

UC Berkeley

UC Berkeley Electronic Theses and Dissertations

Title

So What's the Vibe? Data-Driven Diagnosis of Model Behavior

Permalink

<https://escholarship.org/uc/item/9sj5h376>

ISBN

9798189271427

Author

Dunlap, Lisa

Publication Date

2026-05-01

Peer reviewed|Thesis/dissertation

So What's the Vibe? Data-Driven Diagnosis of Model Behavior

by

Lisa Dunlap

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Joseph E. Gonzalez, Chair

Professor Trevor Darrell

Professor Jacob Steinhardt

Spring 2026

So What's the Vibe? Data-Driven Diagnosis of Model Behavior

Copyright 2026
by
Lisa Dunlap

Abstract

So What’s the Vibe? Data-Driven Diagnosis of Model Behavior

by

Lisa Dunlap

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Joseph E. Gonzalez, Chair

Building a machine learning model is an iterative process: train, evaluate, identify failures, and improve. Benchmarks once anchored this loop, but as models have improved and deployment has expanded, evaluation has become the bottleneck. Metrics fail to capture what practitioners care about, failures are hard to diagnose without manual inspection, and preference signals from users are noisy and hard to decompose. Model development is now constrained not by capability, but by our ability to measure and understand model behavior.

To understand your model, you must understand your data. By systematically analyzing the data that flows through models (their outputs, the real-world conversations users have with them, and the training examples that shaped them) we can characterize what people actually use models for, identify where and why models fail, and understand what drives their behavior.

We develop a suite of methods that operationalize this view across the model development lifecycle. To diagnose output-level differences, we introduce **VisDiff**, which automatically describes differences between image sets in natural language, and **CompCon**, which discovers input-dependent divergences between generative models. To build better evaluation data, we create **Arena-Hard**, a pipeline for curating high-quality benchmarks from crowd-sourced interaction data, and **VisionArena**, a dataset of 230K real-world user–VLM conversations with preference labels. To connect evaluation signals to model behavior, we introduce **VibeCheck**, which discovers and quantifies the qualitative stylistic axes that predict user preferences. Finally, **SAE Data Analysis** shows that sparse autoencoder embeddings provide a unified, interpretable, and cost-effective infrastructure for all of the above tasks, enabling practitioners to trace observed behaviors back to the training data that caused them. Together, these contributions reframe the evaluation bottleneck as a solvable data analysis problem.

To everyone I had the pleasure of meeting during my PhD.

Contents

Contents	ii
List of Figures	vi
List of Tables	xiii
1 Introduction	1
2 Describing Differences in Image Sets with Natural Language	5
2.1 Introduction	5
2.2 Related Works	7
2.3 Set Difference Captioning	8
2.4 Our Method: VisDiff	10
2.5 Results	12
2.6 Applications	14
2.7 Conclusion	18
3 Discovering Divergent Representations between Text-to-Image Models	19
3.1 Introduction	19
3.2 Related Work	20
3.3 Divergent Representation Discovery	22
3.4 ID ² Dataset and LLM Evaluation	26
3.5 Experiments	28
3.6 Conclusion	32
4 From Crowdsourced Data to High-Quality Benchmarks: Arena-Hard and BenchBuilder Pipeline	33
4.1 Introduction	33
4.2 Related works	34
4.3 How do you measure benchmarks?	35
4.4 The BenchBuilder Pipeline and Arena-Hard-Auto v0.1 Dataset	37
4.5 Evaluation with LLM-as-a-Judge	39
4.6 Experimental Results	41

4.7	Limitations	44
4.8	Conclusions	45
5	VisionArena: 230K Real-World User-VLM Conversations	46
5.1	Introduction	46
5.2	Related Works	48
5.3	Dataset and Platform	48
5.4	Data Analysis	50
5.5	Instruction tuning vision-language models	55
5.6	VisionArena-Bench: An Automatic Offline Human-Preference Benchmark for VLMs	56
5.7	Discussion, Limitations, and Future Work	57
6	VibeCheck: Discover & Quantify Qualitative Differences in Large Language Models	59
6.1	Intro	59
6.2	Related Work	60
6.3	Vibe-based Evaluations	61
6.4	VibeCheck	63
6.5	Results	64
6.6	Applications	67
6.7	Limitations	70
6.8	Conclusion	71
7	Interpretable Embeddings with Sparse Autoencoders: A Data Analysis Toolkit	72
7.1	Introduction	72
7.2	Related Work	73
7.3	Methods	74
7.4	Experiments	74
7.5	Case Studies	82
7.6	Limitations & Conclusion	83
8	Conclusion	86
8.1	Lessons Learned	86
8.2	Future Work	87
A	Supplementary Material	89
	Chapter 2: Describing Differences in Image Sets	89
A.1	Supplementary Section 3	90
A.2	Supplementary Section 4	94
A.3	Supplementary Section 5	95

A.4	Supplementary Section 6	98
A.5	Failure Cases and Limitations	104
	Chapter 3: Discovering Divergent Representations	107
A.6	Additional VibeCheck Qualitative Results	108
A.7	Choice of Model and Error Sensitivity	111
A.8	Additional VibeCheck Details	113
A.9	Dataset and Evaluation Details	115
A.10	Experimental Details	118
A.11	Limitations and Failure Cases	119
A.12	Additional Discussion	121
	Chapter 4: Arena-Hard	128
A.13	Appendix	128
A.14	Prompts	131
A.15	Additional Examples	133
	Chapter 5: VisionArena	135
A.16	Interface Details	136
A.17	Topic Distribution	136
A.18	More Data Stats	137
A.19	Category Details	139
A.20	Contamination with Existing Benchmarks	146
A.21	VisionArena-Bench	147
A.22	Additional model details	149
A.23	Failure Cases	149
	Chapter 6: VibeCheck	155
A.24	Experimental Details & Dataset Statistics	155
A.25	Gold standard labels	157
A.26	Generating Preset Vibes	157
A.27	Additional VibeCheck Details	159
A.28	Further Related Works	164
A.29	Limitations	165
A.30	Vibes from each Application	165
A.31	More LLama VS GPT Examples	167
	Chapter 7: Interpretable Embeddings with Sparse Autoencoders	169
A.32	Methods	169
A.33	Additional Related Work	170
A.34	Latent labeling prompts	171
A.35	Additional Results—Dataset Diffing	171
A.36	Additional Results—Correlations	181
A.37	Additional Results—Clustering	188
A.38	Additional Results—Retrieval	191
A.39	Methodological Details from OpenAI Case Study	202
A.40	Ablations on Reader Model Size	203

A.41 Properties of SAE Latents	204
A.42 LLM Judge Details	205
A.43 Dataset Generation Details	208
Bibliography	212

List of Figures

1.1	The evaluation bottleneck. Machine learning development is an iterative cycle of training, deployment, evaluation, diagnosis, and improvement. While training and deployment are increasingly well-supported, the evaluation and diagnosis stages remain the bottleneck: understanding what models are expected to do, where they go wrong, how they compare, and whether we are measuring the right things are open and time-consuming challenges.	1
1.2	Data-driven diagnosis of model behavior. Standard benchmarks indicate <i>that</i> a model differs from another, but not <i>where</i> , <i>why</i> , or <i>what</i> in the data caused it. This thesis develops tools for each layer of this diagnostic stack, progressively deepening the understanding available to a practitioner. Each group addresses one layer; chapter numbers are shown for each contribution.	3
2.1	VisDiff overview. Given two sets of images, VisDiff automatically generates natural language descriptions of what is systematically different between them. .	5
2.2	VisDiff algorithm. VisDiff consists of a <i>GPT-4 proposer</i> on <i>BLIP-2</i> generated captions and a <i>CLIP ranker</i> . The <i>proposer</i> takes randomly sampled image captions from $\mathcal{D}_A$ and $\mathcal{D}_B$ and proposes candidate differences. The <i>ranker</i> takes these proposed differences and evaluates them across all the images in $\mathcal{D}_A$ and $\mathcal{D}_B$ to assess which ones are most true.	6
2.3	Top 5 descriptions generated by the caption-based, image-based, and feature-based proposer. All the top 5 descriptions from the caption-based proposer and the top 2 from the image-based proposer identify the ground-truth difference between “practicing yoga” and “meditating”, while feature-based fails. We report AUROC scores from the same feature-based ranker described in section 2.4.	8
2.4	VisDiff performance under noise. We randomly swap different percentages of images between $\mathcal{D}_A$ and $\mathcal{D}_B$ to inject noise. Results are computed on 50 paired sets in PairedImageSets-Hard. 95% confidence intervals are reported over three runs.	14
2.5	StableDiffusionV2 vs. V1 generated images. For the same prompt, StableDiffusionV2 images often contain more “vibrant contrasting colors” and “art-works placed on stands or in frames”. Randomly sampled images can be found in Figure A.9.	16

2.6	Memorable(top) vs. forgettable(bottom) images. Memorable images contain more “humans”, “close-up views of body part or objects”, and “humorous settings”, while forgettable images contain more “landscapes” and “urban environments”	18
3.1	CompCon overview. CompCon discovers pairs of prompt concepts and visual attributes where two text-to-image models diverge in their outputs.	19
3.2	CompCon overview. We illustrate our approach for discovering diverging visual attributes (top) and diverging prompt descriptions (bottom). Given two text-to-image models and a set of prompts, we use a VLM to identify visual differences. For each diverging attribute, we iteratively refine diverging prompt descriptions by generating candidate prompts $\mathcal{P}_a^{can}$ from the description, classifying them as diverging ($\mathcal{H}^{div}$) or non-diverging ($\mathcal{H}^{non}$). The objective is to maximize the proportion of generated prompts classified as diverging.	22
3.3	ID² creation. Given a diverging prompt description d_{p_a} and diverging visual attribute a , we use an LLM to generate prompt pairs where one of the prompts mentions the diverging visual attribute. Both prompts are then passed to the same text-to-image model to generate image pairs with the visual difference a	26
3.4	ID² example. <i>Top:</i> We show dataset prompts and corresponding generated images, where the second image row depicts the diverging visual attribute. <i>Bottom:</i> We show the ground truth diverging visual attribute and diverging prompt description, along with outputs from our approach and the baselines. Notice that our method produces outputs that better align with the ground truth.	30
3.5	CompCon results comparing PixArt and SD-Lightning. PixArt associates negative emotions / desolation in urban environments with ‘wet streets,’ while SD-Lightning struggles to depict run-down or dilapidated scenes, where PixArt instead conveys ‘decay.’	31
3.6	Finding bias. CompCon discovers both age and racial bias across different text-to-image models. (Left) Age bias in PixArt for prompts about traditional professions. (Right) Racial bias in Stable Diffusion 3.5 for prompts about media professions.	32
4.1	BenchBuilder Pipeline. Starting with a live data source of crowdsourced user prompts, we first cluster their embeddings to form topic clusters. An LLM annotator then assigns quality scores based on the required skills. Clusters with low quality scores are filtered out, and we sample from the remaining high-quality clusters to create a diverse and challenging dataset of benchmark prompts. . . .	37
4.2	Mean score of various topic clusters in descending order. Higher-scoring clusters correlate to challenging topics, while lower scores indicate trivial or ambiguous questions. A more complete topic cluster plot is in Figure A.19f.	40

4.3	Comparison between MT-bench and Arena-Hard-Auto v0.1. The latter offers significantly better separability between models and tighter confidence intervals. GPT-4-0314 has no variance in Arena-Hard-Auto v0.1 because it is the anchor model.	40
4.4	Win-rate of three model pairs (GPT-4-0613 vs Llama-2-70b-chat, Claude-3-Sonnet-20240229 vs Claude-3-Haiku-20240307, and Mistral-Large vs Mixtral-8x7b-Instruct-v0.1) over “quality score”. We randomly sample 50 queries for each quality score 0-7 and bootstrap a win-rate and confidence interval between model pairs on each score interval of 2. We observe a similar trend of win-rate between model pairs becomes increasingly separable as the quality score increases.	41
5.1	VisionArena overview. VisionArena collects 230K real-world conversations between users and vision-language models, with preference labels from human voters.	46
5.2	Interface for anonymous side-by-side chat. Users interact with two anonymous VLMs simultaneously and vote for their preferred response.	49
5.5	Model rankings across question categories and languages. Cells with * have fewer than 100 votes. Certain models achieve a much higher ranking for a particular category, such as InternVL2 on captioning and Chinese, Reka Core on entity recognition, and Llava 1.6 on Humor. Conversely, we see certain model rankings drop, such as Reka Flash on multi-turn and diagrams, and Gemini on Chinese.	53
5.6	Change in model rankings when style control is applied. Cells with * contain battles with fewer than 100 votes. InternVL, Reka Flash, and Llava 1.6 see a ranking drop across the majority of categories, while Claude 3 Haiku and MiniCPM see an increase across most categories. The model lineup changes the most for the captioning, humor, and Vietnamese categories.	53
5.7	VLM failure modes. The top proprietary models fail on questions which require advanced visual reasoning. For example, failure case 1 requires the visual understanding that the cat’s fur patterns looks like another smaller black cat, and the linguistic connection between this and a square root. Non-truncated outputs in Section A.23.	55
6.1	Core components of VibeCheck. A vibe is an axis along which a pair of outputs differ: for example, in the top panel, output A is more friendly while output B is more formal, defining a friendliness vibe. To score a prompt output triplet, a panel of LLM judges are used to determine which output falls higher on the vibe, resulting in a score of 1 (A), -1(B), or 0(tie). Finally, the scores obtained over a large set of outputs along with preference labels are used to compute vibe utility.	62

6.2	Comparing Llama-3-70b VS GPT-4 & Claude-3-Opus on Chatbot Arena. Negative separability scores indicate Llama-3-70B aligns with the low (red) description, while negative preference coefficients show alignment with low descriptions is preferred. We see that Llama is more humorous, utilizes more formatting, provides more examples, and comments much less on ethics than GPT and Claude: all attributes which correlate positively with human preference.	67
6.3	Comparing user preference and separability across STEM and writing tasks. On predefined list of vibes referenced in Table 6.2. Negative preference coefficients indicate a preference for low-description vibes, while negative separability scores show Llama responses align more with the low description than Claude or GPT responses. For writing tasks, detailed explanations, humor, and expressive emotion positively correlate with human preference, while these traits negatively correlate with STEM tasks. Conversely, logical rigor has a stronger positive impact on preference for STEM tasks. These trends are reflected in separability scores, with less separability on STEM tasks for vibes like humor and emotional tone, and more separability for logical rigor.	69
6.4	Top 5 vibes comparing GPT-4o to Llama-3-405B on MATH CoT. Negative separability scores indicate GPT-4o aligns with the low (red) description, while negative preference coefficients show alignment with low descriptions is preferred. GPT-4o outputs contain more LaTeX/MathML formatting which positively correlated with human preference while Llama-3-405B has very structured and overly-detailed responses, which is negatively correlated with preference.	70
7.1	Converting text documents into interpretable embeddings with sparse autoencoders. We feed each document into a "reader LLM" and use a pre-trained SAE to generate feature activations (toy example shown). Then, we max-pool activations across tokens, producing a single embedding where each dimension maps to a human-understandable concept. The interpretable nature of this embedding allows us to perform a diverse range of downstream data analysis tasks.	73
7.2	Average difference of judge-verified frequencies for generated hypotheses. SAEs find bigger differences than the LLM baseline.	77
7.4	SAE embeddings discover novel clusters. On GSM8k answers, dense embeddings <i>[left]</i> and instruction-tuned embeddings <i>[middle]</i> tend to cluster by math problem content. Filtering SAE embeddings to reasoning-related latents creates clusters of various reasoning approaches <i>[right]</i>	80
7.5	MAP averaged over queries, for each method and dataset. Query expansion uses 1–20 phrases; temperature varies from 0.01–1.5.	81

7.6	Emerging characteristics over new generations of OpenAI models. All frequencies shown are judge-verified. Full labels in Section A.39. <i>[Left four]</i> To uncover general changes, we search for and relabel latents with increasing frequencies across generations. We find emerging trends ranging from behavioral to syntactic. <i>[Right]</i> To find changes for <i>specific</i> prompt categories, we extract latent pairs between prompts and their responses that increasingly co-occur over time. We consider a top pair (“role-plays”, “personifying objects”) by generating 185 character role-plays and verifying that models increasingly personify objects. We provide an example on the right.	81
7.7	Triggering the response “I hope it is correct” in Tulu-3. Given five features and the 10k dataset samples, we first verify that math prompts which contain “I hope it is correct” in the response have these features <i>[left]</i> . Then, we generate responses from Tulu-3 on new prompts varying along the five feature axes <i>[right]</i> . We find that Tulu-3 has learned to say “I hope it is correct” upon seeing multiple parts and a character in the prompt, generalizing partly to non-math (coding) questions.	83
7.8	Identification and investigation of spurious correlation in Tulu-3’s SFT dataset. Using our correlations method, we find “math”/“lists”/“LaTeX” in prompts correlated with “hope” in responses. Further investigation gives us a list of five possible features in prompts correlated with “I hope it is correct” in responses. Has the model learned to say this, and under what kinds of prompts?	85
A.1	PairedImageSets examples where VibeCheck succeeds. We show the ground-truth difference, top difference predicted by VibeCheck, AUROC score output by the ranker, and evaluation of the predicted difference by GPT-4.	92
A.2	Prompt used to generate paired sentences for VisDiffBench (top) and evaluate VisDiffBench (bottom). Input-dependent texts are colored in blue.	93
A.3	Example input to the image-based proposer. We arrange 20+20 input images into a single 4-row, 10-column gridded image.	94
A.4	Prompt for caption-based proposer (top), image-based proposer (middle), and caption-based ranker (bottom). Input-dependent texts are colored in blue. . . .	96
A.5	Analysis of the number of rounds (left) and number of samples (right) for the proposer on 50 PairedImageSets-Hard sets. 3 rounds and 20 samples are the best in terms of performance and efficiency.	97
A.6	ImageNetV2 vs. ImageNet. All V2 images are shown in the first row while V1 images are shown in the second row. We show the class name and top 3 difference descriptions generated by VibeCheck.	99
A.7	CLIP vs. ResNet. All CLIP correctly classified but ResNet incorrectly classified images are shown in the first row while other images are shown in the second row. We show the class name and top 3 difference descriptions generated by VibeCheck.	100
A.8	Modified proposer’s prompt for StableDiffusion analysis.	101

A.9	Randomly sampled images generated from SDv2 and v1 using DiffusionDB prompts.	103
A.10	PairedImageSets examples where VibeCheck fails. We show the ground-truth difference, top difference predicted by VibeCheck, AUROC score output by the ranker, and evaluation of the predicted difference by GPT-4.	105
A.11	Further qualitative results comparing PixArt-Alpha to SDXL-Lightning using a templated prompt bank.	109
A.12	Qualitative results comparing PixArt-Alpha to SDXL-Lightning using an LLM generated initial prompt bank.	110
A.13	Diverging attributes discovered by CompCon across four models. Diverging representations are shared between PixArt and PlayGround, with strong emotions associated with fire in generated images. Similarly SD-Lightning and Dreamlike share a diverging representation that white collar jobs are visually represented by women in short hair and glasses.	111
A.14	Example image grid input to VLM during the diverging visual attribute discovery.	123
A.15	Example input-dependent difference in ID ²	124
A.16	Example of CLIP classifying prompts as diverging when both (left) or neither (left) image contains the attribute.	125
A.17	Finding bias in PixArt-Alpha and SDXL-Lightning.	126
A.18	VibeCheck results comparing PixArt and SD-Lightning over initial and final iterations. Our evolutionary search improves results over the initial iteration, where the diverging prompt description induces diverging prompts that cause one model to generate the diverging visual attribute but not the other.	127
A.20	Interface for anonymous side-by-side chat.	136
A.21	Top 20 topic clusters of VisionArena-Chat compared to WildVision-Chat. VisionArena-Chat includes more diverse and broad topics especially in the STEM field.	137
A.26	Proportion of Refusals per model.	140
A.27	VisionArena-Battle Model Ranking Results.	141
A.28	Random Samples from VisionArena-Battle	142
A.29	Random Samples from VisionArena-Chat	143
A.30	VisionArena-Battle category overlap.	144
A.31	Hard OCR Failure Cases. These failure cases show that models still fail in cases where the text is perturbed such as rotations or messy hand-writing.	151
A.32	Hard Counting Failure Cases. These cases show that proprietary models still fail at counting tasks involving shapes even when the format is clear.	152
A.33	Hard Reasoning Failure Cases (part 1). The failure cases highlight the model's inability to connect the visual reasoning with language reasoning. While the vision model identifies an object, it is not able to identify the relationship between them correctly.	153

A.34 Hard Reasoning Failure Cases (part 2). These failure cases highlight the inability for the model to be able to correctly map out a grid-like structure and the various pieces in it.	154
A.42 Supreme Leader example from Chatbot Arena	168
A.43 Detailed methodology for each of the four tasks.	169
A.44 Verification rates of generated hypotheses for diffing. We find that SAEs generate valid hypotheses more often than our LLM baselines when comparing multiple models (left three) and similarly otherwise (right three).	175
A.45 Coverage of generated hypotheses overall. We compute the % of responses that have at least one hypothesis with the "target" dataset uniquely verified. The generated hypotheses for SAEs have greater coverage of the unique qualities of target datasets over pure LLMs on multi-model setups (left three). SAEs have similar or slightly worse coverage for two-model cases (right three).	176
A.46 Histogram of correlation metric (left: NPMI, right: CO) and semantic similarity of latent pairs. We choose NPMI as our metric.	182
A.47 (a)-(d) We plot the discovered group of pairs ($\text{NPMI} > 0.8$, semantic similarity < 0.2) for each type of text injected, with 0.5% of texts being injected texts. Relevant pairs are colored. (e) We show the proportion of relevant pairs in the candidate group for different injection levels 0.1%-1%. (f) We inject all 3 texts at once.	182
A.48 CDF of conditional occurrence for pairs discovered by every method, for Civil-Comments (left) and the Pile (right).	184
A.49 Dense embedding (top row), instruction-tuned embedding (middle row) and SAE embedding (bottom row) clustering results: (1) topic (2) sentiment (3) temporal framing and (4) writing style. Mappings from clusters to true labels are chosen with the Hungarian algorithm [135].	189
A.51 Per-cluster accuracies for different n_{clusters} for prompts, responses and the Pile. The solid lines are the median, dashed lines the interquartile range and dotted lines the range.	190
A.53 MP@50 averaged over queries for each method and dataset. Query expansion uses 1–20 phrases; temperature varies from 0.01 to 1.5.	194
A.56 Ranking similarity among the relevant documents, using Rank-Biased Overlap (RBO) [275] with hyperparameter $p = 0.98$ since we are concerned about the top 50 results.	196
A.57 F1 scores after relabeling SAE latents per dataset.	204
A.58 F1 scores using fixed SAE latent labels (based on LMSYS-1M).	204
A.59 Left: Empirical CDF of normalized average precision of the classifier for latents in each log-frequency range. Right: Automatic interpretability score summary.	204

List of Tables

2.1	Summary of VisDiffBench. In experiments, we merge ImageNetR and ImageNet* because they have limited sets.	9
2.2	Results on VisDiffBench. GPT-4V image-based and BLIP-2 caption-based proposers with CLIP feature-based rankers consistently outperform other proposers and rankers by a large margin. We use the caption-based proposer with the CLIP ranker as the final VisDiff algorithm because it obtains the highest accuracy on the PairedImageSets-Hard and is cheaper than the GPT-4V image proposer.	10
2.3	Top per-class differences between ImageNet and V2.	15
2.4	Top per-class differences between CLIP and ResNet. Acc_C/Acc_R : CLIP/ResNet accuracy.	15
2.5	Accuracy on images with / without people.	17
3.1	Visual attribute and prompt description scores on ID². Our approach outperforms all baselines on both diverging visual attribute and diverging prompt description discovery.	29
4.1	First row is the percentage of queries with quality scores of the column or more in 75K Chatbot Arena data assigned by GPT-3.5-Turbo. Second row is the percentage of queries in 75K Chatbot Arena labeled by GPT-3.5-Turbo with each of the 7 qualities.	41
4.2	We use a set of top-20 models ¹ on Chatbot Arena (2024/04/13) that are also present on the AlpacaEval leaderboard to calculate separability and agreement per benchmark. We consider the human preference ranking by Chatbot Arena (English only) as the reference to calculate agreement.	42
4.3	Statistics of Arena-Hard-Auto v0.1 with four LLM different judges: GPT4-T (gpt-4-1106-preview), Claude-3-Opus, Gemini1.5-Pro (gemini-1.5-pro-0514), Llama3-70B (llama-3-70b-instruct). We compare rankings produced by these judges against Chatbot Arena (English) ranking (as of 2024/04/13). We observe GPT-4T and Gemini1.5-Pro have higher agreement than Claude-3-Opus and Llama-3-70B. Furthermore, the ensemble of GPT4-T and Gemini1.5-Pro shows even higher agreement.	42

4.4	We compare Arena-Hard-Auto v0.1 with two sets of 500 prompts randomly sampled from 75K Chatbot Arena user queries. We evaluate the set of top-20 models and compare various statistics across. Each prompt is judged only once by positioning the baseline answer first.	43
4.5	We compare Arena-Hard-Auto v0.1 (gpt-4-1106-preview as judge) to Chatbot Arena Category Hard Prompt (English) on the same set of top-20 models*. By comparing Arena-Hard-Auto v0.1 to a similar distribution of queries from Chatbot Arena, we obtain even higher alignment to human preferences.	43
5.1	Dataset Comparison. Compared to previous VLM preference benchmarks, VisionArena contains 3x the amount of data, with more users, language, models, unique images, and conversation turns.	47
5.2	Percentage of ‘tie (bothbad)’ per category. Captioning, creative writing, and humor categories have low percentages of bothbad responses compared to coding and homework problems.	54
5.3	Performance across models trained with different instruction tuning datasets. Llama-3.2-11B-Vision-Instruct scores are author-reported. Fine-tuning on samples from VisionArena-Chat outperforms fine-tuning on Llava-Instruct across all benchmarks. Llama-3.2-VisionArena also outperforms Llama-3.2-11B-Vision-Instruct on both MME and WV-Bench, despite being fine-tuned on 30x less data.	56
5.4	Correlation of rankings with ChatbotArena’s VLM leaderboard. Performance comparison on 16 models ² between VisionArena-Bench to Chatbot Arena’s VLM leaderboard based on confidence agreement, spearman correlation, and Kendall tau correlation. Result as of leaderboard on October 23, 2024. . . .	57
6.1	Comparison of VibeCheck vibes to human labels. Complete table in Figure A.36a. We see that the vibes discovered by VibeCheck closely align with vibes found through human analysis.	65
6.2	Comparing Llama-3 to GPT and Claude on Chatbot Arena. We report Model Matching Accuracy (M.M.), Preference Prediction Accuracy (P.P.), and average Cohen’s Kappa (C.K) for the full dataset (Overall) and STEM and Writing categories. VibeCheck achieves higher model matching accuracy than Predefined Vibes and similar preference prediction accuracy. VibeCheck obtains the largest improvements over predefined vibes in the writing category, suggesting that for open-ended prompts, model styles differ significantly, and style has a greater influence on preference.	68
7.1	Latent with the biggest frequency difference for tone changes (left) and movie genre differences (right). Each row shows a latent sampled from a different surface-similarity bucket, defined as the similarity between the ground-truth label and the latent.	76

7.2	Qualitative examples of differences between model behaviors. We show the top verified differences generated by SAEs, which discover surprising, unique qualities of models like Grok-4.	76
7.3	Token usage by SAEs and baselines. SAEs take 2–8× fewer tokens to generate differences. Breakdowns in Table A.25.	77
A.1	Paired sentences for PairedImageSets. Easy, medium, and hard examples are shown in the left, middle, and right.	91
A.2	Correlation of automated metric with humans. Model-free metrics include B4 (BLEU-4), R1 (ROUGE-1), RL (ROUGE-L). Model-based metrics include BS (BERTScore), V1.5 (Vicuna-1.5-13B), G3.5 (gpt-3.5-turbo-0613), G4 (gpt-4-0613).	94
A.3	VisDiffBench results using different ranking metrics based on CLIP similarity scores. We use the caption-based proposer. A1 & A5 are Acc@1 & Acc@5.	95
A.4	Results on VisDiffBench with different captions and language models. We bold any numbers within 0.02.	97
A.5	Concepts more true for SDv2 than v1. Differences are proposed by running VibeCheck on PartiPrompts images. These differences obtain similar scores on the unseen DiffusionDB images, indicating that these differences generalize to various prompts.	102
A.6	Top 25 differences for memorable and forgettable images.	104
A.7	Art Styles, Adjectives, and Subjects for template prompts. "{art style} of a {adjective} {subject}"	108
A.8	Proportion of Diverging Prompts per iteration. Notice that running VibeCheck for more iterations leads to a description which produces a higher proportion of diverging prompts.	112
A.9	CompCon performance using different VLMs/LLMs.	112
A.10	VisionArena Ground Truth splits.	120
A.11	VisionArena Leaderboard (baseline: GPT-4-0314). *GPT-4-Turbo’s high score can be due to the GPT-4 judge favoring GPT-4 outputs.	128
A.12	Length Bias Comparison Between GPT and Claude as Judge. Prompt for detailed: “You are a helpful assistant who thoroughly explains things with as much detail as possible.”, prompt for chatty: “You are a helpful assistant who is chatty.”, and default prompt: “You are a helpful assistant.”	129
A.13	Variances between 3 separate runs of VisionArena on gpt-3.5-turbo’s performance using GPT-4-1106-Preview as judge.	129
A.14	Comparing Gemini-1.5-Flash question categorization to larger models. Gemini-1.5-Flash is evaluated against SOTA models on 1000 samples from VisionArena-Chat, using Gemini-1.5-Pro for image-based prompts and Claude-3.5-Sonnet for text-based prompts. Gemini-1.5-Flash achieves high agreement with SOTA models for category classification.	145

A.15 Proportion of benchmark data in VisionArena-Chat.	147
A.16 VisionArena-Bench leaderboard (baseline: GPT-4-Turbo)	148
A.17 VisionArena-Bench leaderboard (baseline: claude-3-sonnet-20240229)	149
A.18 Model Name to exact model version	150
A.19 Model performance across several hard tasks. O indicates that the model solves the problem and X indicates that the model fails to solve the problem. 9 out of 16 models fail all questions.	150
A.20 Dataset Statistics	155
A.21 Model Win Rates	155
A.22 VibeCheck Hyperparameters	156
A.23 Predefined vibes. We prompt GPT-4o to generate a set of 10 vibes which represent common axes on which LLM outputs differ.	158
A.24 Datasets used for ground-truth evaluation in data diffing.	173
A.25 Token usage per model when generating hypotheses for comparing datasets.	177
A.26 Generated hypotheses using SAE embeddings.	178
A.27 LLM-S diffing hypotheses (generate differences and summarize).	179
A.28 LLM-C diffing hypotheses (generate differences and cluster).	180
A.29 Keywords used to judge if a latent pair is relevant to the injected correlations.	182
A.30 For each type of injected correlation, at various injection rates, we count the number of batches where the LLM correctly identifies a related correlation.	183
A.31 Hypothesized correlations in CivilComments generated by LLM.	185
A.32 Hypothesized correlations in the Pile generated by LLM.	186
A.33 Sample of 20 Tulu hypotheses generated by LLM.	187
A.34 Example clusters from ChatbotArena prompts.	190
A.35 Example clusters from ChatbotArena responses.	191
A.36 Example clusters from The Pile.	191
A.37 Example queries across the six datasets.	192
A.38 Baselines used for the property-based retrieval benchmark.	196
A.39 For the OpenAI+LLM and SAE methods, we fix the hyperparameters to be their best values averaged across datasets ($n_{\text{phrases}} = 18$ and $T = 0.2$), and report their individual and combined performance per dataset. We also add in LLM reranking of the top 50.	197
A.40 Comparison of top 3 retrieval results for OpenAI, OpenAI+LLM and SAE methods, for the “model stuck in repetitive loop” query.	197
A.41 Comparison of top 3 retrieval results for OpenAI, OpenAI+LLM and SAE methods, for the “model shows its reasoning” query.	198
A.42 ChatbotArena prompts: Top 3 most improved and most degraded queries.	199
A.43 ChatbotArena responses: Top 3 most improved and most degraded queries.	199
A.44 Reasoning traces: Top 3 most improved and most degraded queries.	200
A.45 The Pile: Top 3 most improved and most degraded queries.	200
A.46 Biology abstracts: Top 3 most improved and most degraded queries.	201
A.47 Short stories: Top 3 most improved and most degraded queries.	201

A.48 Sample of latents that are most (top decile) and least (bottom decile) predictable
by NAP in each frequency bin with autointerp scores $\geq$ 70% (i.e. “good” latents). 206

A.49 Vibes and their associated system prompts. 209

Acknowledgments

Throughout my PhD I have been astounded by just how truly amazing everyone I have met is: whether it be helping with research, planning and executing shenanigans, or being a shoulder to cry on, I will always be grateful that any of you put up with me at all. This dissertation exists because of your patience, and despite my best efforts.

To my family and family friends: thank you for your support, and for having absolutely no chill. Watching you spend your time cycling around the world, getting pilot’s licenses, earning PhDs, or casually adding an MD-PhD (flex) really peer pressured me into success.

To my undergraduate mentors, Richard Liaw and Alvin Wan: I wouldn’t have started research without you. I’m still not sure whether recruiting me was a net positive for the field, but it certainly changed my life for the better, so thank you.

To my advisors, Joey, Jacob, and Trevor: thank you for putting up with all my shenanigans, and for your patience during the many months when I had no results because I decided to take a detour into web development. Trevor, thank you for taking a chance on a systems admit who wanted to break into AI. Jacob, thank you for letting me sneak into your group a few years into the PhD and cosplay as your student for four years. Joey, thank you for shepherding me from a scared 19-year-old who couldn’t index a NumPy array to a slightly less scared 27-year-old who has since learned to just use PyTorch.

To the mentors and professors beyond my committee, Alyosha, Angjoo, Jitendra, Aditi, Matei, Ion, Sarah, Bryan, Joseph, Fabian, Lance, and many others: thank you for your literal and figurative open doors, for the hikes, the post-deadline drinks, and the incredibly unproductive but amusing group meetings.

To my undergraduate mentees and coauthors: Clara, Paris, Alyssa, Elva, Tim, Chris, Evan, Terry, Naz, Ethan, Nick, Krisha, Karina, Shreyas, Stephen, Esau, Socrates, Mark, and Pratyay. Thank you for the late nights, the willingness to run “just one more experiment,” and for turning my incoherent Slack messages into actual papers. Working with you has been one of the most rewarding parts of my PhD, and I can’t wait to see what you all do next.

Finally, to my friends and peers who made me excited to come to work every day, including but certainly not limited to: Conor, Laura, Suzie, Shishir, Chung-min, Grace, David, Brent, Parth, Alan, Mir, Patrick, Paras, Charles, Sarah, Daniel, Kevin, Alex, Shaddaj, Devin, Medhini, Ruiqi, Yihui, Anja, Haven, Fiona, and Ritwik. Yall made the hard days manageable and the good days unforgettable, whether through 2 a.m. AI doomer conversations, workshopping paper titles over Sweetgreen, or reminding me that there is, in fact, a world outside the lab.

And to my dear friend Sam, who passed away midway through my PhD: you were there from my very first day at Berkeley and were my biggest cheerleader all the way through, even on the many occasions when I was clearly wrong.

Chapter 1

Introduction

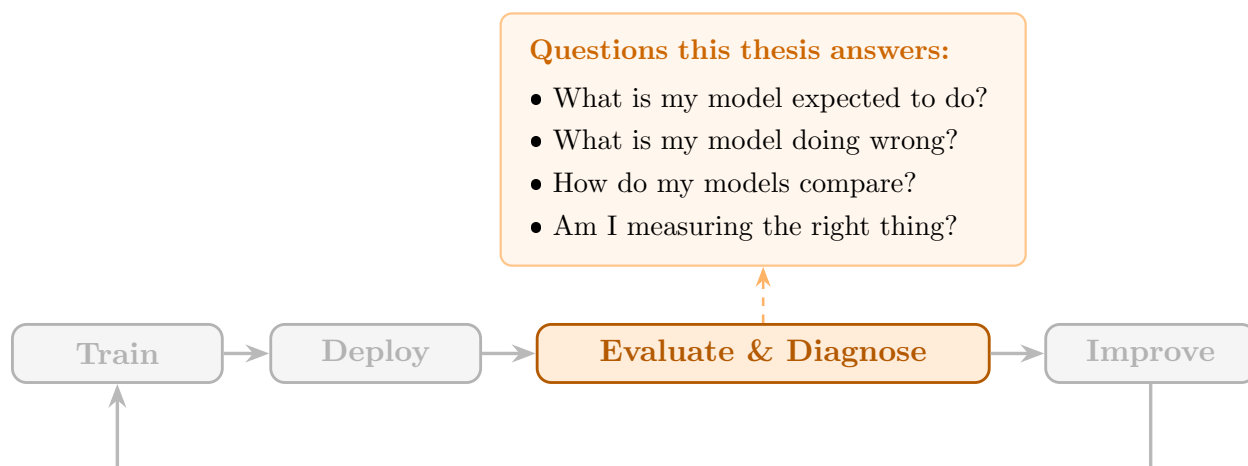


Figure 1.1: **The evaluation bottleneck.** Machine learning development is an iterative cycle of training, deployment, evaluation, diagnosis, and improvement. While training and deployment are increasingly well-supported, the evaluation and diagnosis stages remain the bottleneck: understanding what models are expected to do, where they go wrong, how they compare, and whether we are measuring the right things are open and time-consuming challenges.

Building a machine learning model is an iterative process: train a model, evaluate its performance, identify failures, and improve. For much of the field’s history, this loop was well-supported by benchmarks, curated datasets that provided a reliable signal for measuring progress and comparing methods. Benchmark-driven development produced some of the most significant advances in the field, enabling models to achieve strong performance across tasks from image classification [55] to language understanding [267] to code generation [34].

As models have improved and the tasks they are deployed on have expanded, however, the evaluation step has become the bottleneck. Practitioners routinely find that their metrics do not fully capture what they care about, that failures are difficult to diagnose without

manually inspecting outputs, and that preference signals from real users are noisy and hard to decompose. The result is that model development is increasingly constrained not by capability, but by our ability to measure and understand what our models are doing.

The problem has three distinct layers. The first is *uninformative metrics*: standard benchmarks tell you *that* your model is failing but not *where* or *why*: a number going down gives no foothold for improvement. The second is *unrepresentative benchmarks*: benchmarks are curated ahead of time and quickly diverge from the tasks users actually bring to deployed models, so a model can score well on every benchmark while failing at what matters. The third is *intractable inspection*: even when practitioners have representative data and suspect something is wrong, the scale of modern systems makes manual diagnosis impractical. There is no principled, automated way to characterize what kinds of inputs cause failures, how one model’s behavior differs from another’s, or what in the training data explains an observed behavior.

Knowing that a model’s accuracy dropped from 75% to 72% tells you something has changed, but not what, or why, or how to fix it. Knowing users prefer Model A over Model B 62% of the time raises an obvious follow-up: what is Model A doing that Model B is not? Knowing a model scores well on every benchmark is useful until you discover that users are asking multi-turn spatial reasoning questions and open-ended creative tasks that no benchmark tested.

The answers to these questions live in the data: the model’s outputs, the real conversations users have with it, and the training examples that shaped its behavior. Systematic differences between two models’ outputs can often be described in natural language once you look. Real-world interaction data reveals the task distributions practitioners actually face, not the ones researchers assumed. And the specific phrases, styles, and tendencies a model exhibits can often be traced to clusters of documents in its training corpus. The chapters in this thesis each address one of these layers as a data problem: what does the data tell us about where a model fails, what kinds of inputs users bring to it, and why it behaves the way it does. The work spans both vision and language models (the data problems are the same regardless of modality) and is organized into three groups, summarized in Figure 1.2.

Diagnosing model failures across inputs and outputs. When a model underperforms, the raw signal (a number going down, a preference vote lost) tells you nothing about what went wrong. The data does. The most direct version of the problem is: given two sets of model outputs or two datasets, automatically produce a human-readable description of what is systematically different between them. We introduce two methods for this, starting with visual data and extending to generative models.

VisDiff (Chapter 2) formalizes *set difference captioning*: given two sets of images, generate natural language descriptions of what distinguishes them. This turns a vague question (“why does my model fail more on this subset?”) into a concrete, actionable answer. VisDiff uses a proposer-ranker framework in which candidate descriptions are generated from image captions and re-ranked by how well they discriminate between the two sets.

So What’s the Vibe? Data-Driven Diagnosis of Model Behavior		
■ <i>Introduction</i>		
Ch. 1	Introduction	Motivates the evaluation bottleneck and provides an overview of contributions
■ <i>Diagnosing model failures across inputs and outputs</i>		
Ch. 2	VisDiff	Describes what is <i>systematically different</i> between two image sets in natural language
Ch. 3	CompCon	Discovers which prompt concepts <i>trigger</i> divergent visual attributes between generative models
■ <i>Understanding real-world model usage via Chatbot Arena</i>		
Ch. 4	Arena-Hard	Identifies Chatbot Arena prompts that <i>most discriminate</i> between models and builds reliable, cost-effective offline benchmarks
Ch. 5	VisionArena	Reveals <i>what tasks users actually bring</i> to Chatbot Arena and where vision-language models fail, across 230K real conversations
■ <i>Understanding model behavior and building interpretable analysis tools</i>		
Ch. 6	VibeCheck	Discovers and quantifies qualitative “ <i>vibes</i> ” (stylistic axes like thoroughness and tone) that predict user preferences
Ch. 7	SAE Analysis	Interpretable embeddings for dataset diffing, correlation discovery, clustering, and retrieval at 2–8× lower cost than LLM approaches
■ <i>Conclusion</i>		
Ch. 8	Conclusion	Synthesizes recurring themes, lessons learned, and open problems

Figure 1.2: **Data-driven diagnosis of model behavior.** Standard benchmarks indicate *that* a model differs from another, but not *where*, *why*, or *what* in the data caused it. This thesis develops tools for each layer of this diagnostic stack, progressively deepening the understanding available to a practitioner. Each group addresses one layer; chapter numbers are shown for each contribution.

CompCon (Chapter 3) extends this to generative models, where the question is not just *what* is different in existing outputs but *what inputs trigger* the difference. CompCon uses an evolutionary search algorithm to discover pairs of prompt concepts and visual attributes where two text-to-image models diverge; for example, one model depicts “loneliness” with

wet streets while the other does not.

Understanding real-world model usage via Chatbot Arena. Diagnosing differences is only useful if the data reflects what models are actually asked to do. Curated benchmarks don’t: they are designed before deployment and quickly diverge from real user behavior. The richest source of evaluation signal is the data users generate when interacting with models in the wild. The next two chapters build and analyze that data through the Chatbot Arena platform (Chapters 4 and 5).

Arena-Hard (Chapter 4) addresses the curation problem: not all real-world prompts are equally diagnostic. BenchBuilder is a pipeline that identifies the subset of crowdsourced prompts that most reliably discriminates between models, achieving 89% agreement with human preference rankings at a fraction of the cost of live evaluation. It also introduces style control, a method for disentangling model quality from superficial stylistic factors such as response length and formatting.

VisionArena (Chapter 5) extends this infrastructure to vision-language models, collecting 230K real-world user-VLM conversations with preference labels. It surfaces failure modes and task distributions that no curated benchmark anticipated: users ask spatial reasoning questions, multi-turn planning tasks, and style-dependent creative tasks far more than existing benchmarks reflect.

Understanding model behavior and building interpretable analysis tools. Collecting better data and diagnosing output differences are necessary but not sufficient. The data can also tell us *why* models behave the way they do: what stylistic patterns drive user preferences, and what concepts in the training corpus explain observed behaviors.

VibeCheck (Chapter 6) discovers and quantifies the qualitative stylistic axes, which we call “vibes”, that drive user preferences between LLMs. By iteratively proposing candidate vibes from model outputs and validating them with a panel of LLM judges, VibeCheck turns a preference signal (“users prefer Model A 62% of the time”) into an explanation (“Model A is perceived as more thorough; Model B as punchier”).

SAE Data Analysis (Chapter 7) uses interpretable embeddings for data analysis. Rather than analyzing model outputs with expensive LLM calls or opaque dense embeddings, this chapter uses sparse autoencoder (SAE) activations to create interpretable embeddings of text, where each dimension maps to a human-understandable concept. These embeddings support the same diagnostic tasks as prior chapters, including dataset diffing, correlation discovery, clustering, and property-based retrieval, at 2–8 $\times$ lower cost than LLM-based approaches and with greater transparency and controllability than dense embeddings.

Chapter 2

Describing Differences in Image Sets with Natural Language



Figure 2.1: **VisDiff overview.** Given two sets of images, VisDiff automatically generates natural language descriptions of what is systematically different between them.

2.1 Introduction

What kinds of images are more likely to cause errors in one classifier versus another [75, 121]? How do visual concepts shift from a decade ago to now [132, 210, 304]? What types of images are more or less memorable for humans [116]? Answering these questions can help us audit and improve machine learning systems, understand cultural changes, and gain insights into human cognition.

Although these questions have been independently studied in prior works, they all share a common desideratum: discovering differences between two sets of images. However, discovering differences in many, potentially very large, sets of images is a daunting task for humans. For example, one could gain insights into human memory by discovering systematic differences between memorable images and forgettable ones, but finding these differences may require scanning through thousands of images. An automated solution would be more scalable.

In this work, we explore the task of describing differences between image sets, which we term *Set Difference Captioning* (Figure 2.1). Specifically, given two sets of images $\mathcal{D}_A$ and

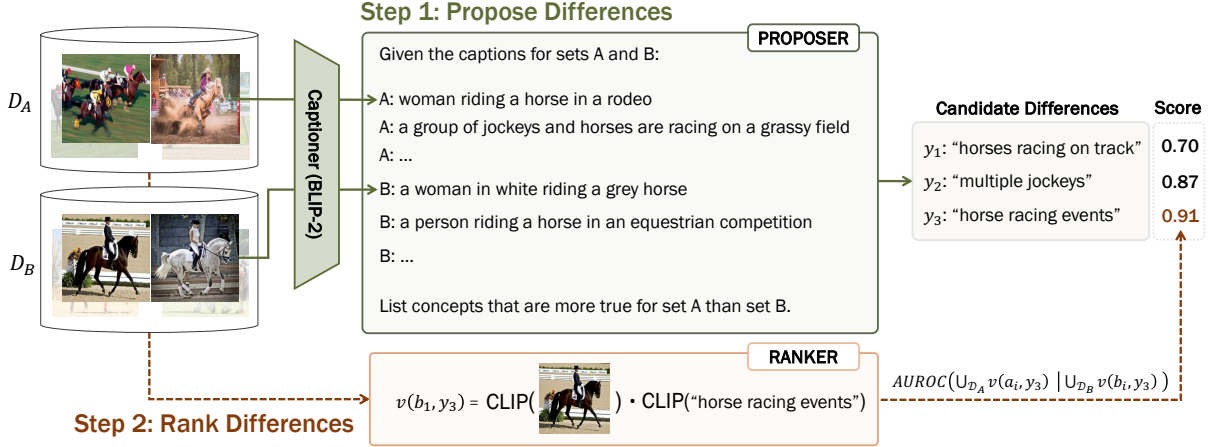


Figure 2.2: **VisDiff algorithm.** VisDiff consists of a *GPT-4 proposer* on *BLIP-2* generated captions and a *CLIP ranker*. The *proposer* takes randomly sampled image captions from $\mathcal{D}_A$ and $\mathcal{D}_B$ and proposes candidate differences. The *ranker* takes these proposed differences and evaluates them across all the images in $\mathcal{D}_A$ and $\mathcal{D}_B$ to assess which ones are most true.

$\mathcal{D}_B$, set difference captioning aims to find the most salient differences by generating natural language descriptions that are more often true in $\mathcal{D}_A$ than $\mathcal{D}_B$. We show in [section 2.6](#) that many dataset and model analysis tasks can be formulated in terms of set difference captioning, and methods that address this problem can help humans discover new patterns in their data.

Set difference captioning presents unique challenges to current machine learning systems, since it requires reasoning over all the given images. However, no existing models in the vision and language space can effectively reason about thousands of images as input. Furthermore, while there are usually many valid differences between $\mathcal{D}_A$ and $\mathcal{D}_B$, end users are typically interested in what can most effectively differentiate between the two sets. For example, “birthday party” is a valid difference in [Figure 2.1](#), but “people posing for a picture” better separates the sets.

We introduce a two-stage proposer-ranker approach [296, 298, 304] for set difference captioning that addresses these challenges. As shown in [Figure 2.2](#), the *proposer* randomly samples subsets of images from $\mathcal{D}_A$ and $\mathcal{D}_B$ to generate a set of candidate differences in natural language. The *ranker* then scores the salience and significance of each candidate by validating how often this difference is true for individual samples in the sets. Within the proposer-ranker framework, there are many plausible design choices for each component, and in this work we investigate three categories of proposers and rankers that utilize different combinations of models pre-trained with different objectives.

To evaluate design choices, we construct VisDiffBench ([Figure 2.3](#)), a dataset consisting of 187 paired image sets with ground-truth differences. We also propose a large language model-based evaluation to measure correctness. By benchmarking different designs on VisDiffBench, we identify our best algorithm, VisDiff, which combines a proposer based on

BLIP-2 captions and GPT-4 with a ranker based on CLIP features. This method accurately identifies 61% and 80% of differences using top-1 and top-5 evaluation even on the most challenging split of VisDiffBench.

Finally, we apply VisDiff to a variety of applications, such as finding dataset differences, comparing model behaviors, and understanding questions in cognitive science. VisDiff identifies both differences that can be validated by prior works, as well as new findings that may motivate future investigation. For example, VisDiff uncovers ImageNetV2’s temporal shift compared to ImageNet [55, 216], CLIP’s strength in recognizing texts within images compared to ResNet [100, 213], StableDiffusionV2 generated images’ stylistic changes compared to StableDiffusionV1 [223], and what images are more memorable by humans [115]. These results indicate that the task of set difference captioning is automatic, versatile, and practically useful, opening up a wealth of new application opportunities for future work and potentially mass-producing insights unknown to even experts across a wide range of domains.

2.2 Related Works

Many prior works explored difference captioning [5, 145, 146, 280] and change captioning [29, 131, 203], which aim to describe differences between *a single pair of images* with language. Recent large visual language models (VLMs) like GPT-4V [193] have shown promise in describing differences in *small groups of images*. However, the question of how to scale this problem to sets containing *thousands of images* remains unanswered. Meanwhile, some existing works in vision tackle understanding large amounts of visual data through finding concept-level prototypes [61, 259], “averaging” large collections of images [303], using simple methods like RGB value analysis [174, 257], or using a combination of detectors and classifiers to provide dataset level statistics [268]. However, they do not describe the differences in natural language, which is flexible and easy to interpret.

Our work draws inspiration from D3 [296] and D5 [298] frameworks, which use large language models (LLMs) to describe differences between text datasets. A recent work GS-CLIP [304] applied a similar framework as D3 in the image domain, using CLIP features to retrieve differences from a pre-defined text bank. While this work targets the task of set difference captioning, it struggles at generating descriptive natural language and has a limited evaluation on the MetaShift [154] dataset that we found contains a significant amount of noise. Inspired by D3 [296], our study advances a proposer-ranker framework tailored for the visual domain, leveraging large visual foundation models and a well-designed benchmark dataset. The versatility and effectiveness of our approach are further demonstrated through applications across a variety of real-world scenarios, underscoring its potential impact and utility in practical settings.

Lastly, the set difference captioning setting is closely related to the field of explainable computer vision. Traditional explainable computer vision methodologies have predominantly concentrated on interpreting features or neurons within deep neural networks, as exemplified

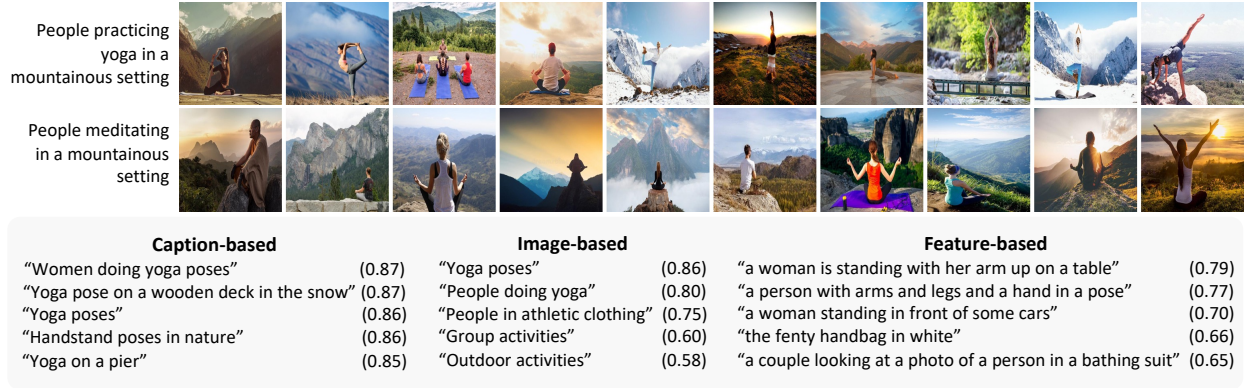


Figure 2.3: **Top 5 descriptions generated by the caption-based, image-based, and feature-based proposer.** All the top 5 descriptions from the caption-based proposer and the top 2 from the image-based proposer identify the ground-truth difference between “practicing yoga” and “meditating”, while feature-based fails. We report AUROC scores from the same feature-based ranker described in [section 2.4](#).

by approaches like LIME [221], CAM [300], SHAP [170], and MILAN [107]. Recent shifts towards a data-centric AI paradigm have sparked a wave of research focusing on identifying influential data samples that impact predictions [205, 236], and on discerning interpretable data segments [46, 57, 75], thereby elucidating model behaviors. Our set difference captioning aligns with these objectives, offering a unique, purely data-driven approach to understanding and explaining differences in image sets with natural language.

2.3 Set Difference Captioning

In this section, we first describe the task of set difference captioning, then introduce VisDiffBench, which we use to benchmark performance on this task.

Task Definition

Given two image datasets $\mathcal{D}_A$ and $\mathcal{D}_B$, the goal of *set difference captioning (SDC)* is to generate a natural language description y that is more true in $\mathcal{D}_A$ compared to $\mathcal{D}_B$. For example, in [Figure 2.3](#), both $\mathcal{D}_A$ and $\mathcal{D}_B$ contain images of horses, but the images from $\mathcal{D}_A$ are all from racing events, so a valid choice of y would be “horse racing events”.

In our benchmarks below, we annotate $(\mathcal{D}_A, \mathcal{D}_B)$ with a ground truth y^* based on knowledge of the data-generating process. In these cases, we consider an output y to be correct if it matches y^* up to semantic equivalence (see [section 2.3](#) for details). In our applications ([section 2.6](#)), we also consider cases where the ground truth y^* is not known.

Dataset	# Paired Sets	# Images Per Set
ImageNetR (sampled)	14	500
ImageNet* (sampled)	23	500
PairedImageSets (Easy/Medium/Hard)	50/50/50	100/100/100

Table 2.1: **Summary of VisDiffBench.** In experiments, we merge ImageNetR and ImageNet* because they have limited sets.

Benchmark

To evaluate systems for set difference captioning, we construct VisDiffBench, a benchmark of 187 paired image sets each with a ground-truth difference description. To create VisDiffBench, we curated a dataset PairedImageSets that covers 150 diverse real-world differences spanning three difficulty levels. We supplemented this with 37 differences obtained from two existing distribution shift benchmarks, ImageNet-R and ImageNet*. Aggregate statistics for VisDiffBench are given in Table 2.1.

ImageNet-R: ImageNet-R [102] contains renditions of 200 ImageNet classes across 14 categories (e.g., art, cartoon, painting, sculpture, sketch). For each category, we set y^* to be the name of the category, $\mathcal{D}_A$ to be 500 images sampled from that category, and $\mathcal{D}_B$ to be 500 original ImageNet images sampled from the same 200 classes.

ImageNet*: ImageNet* [260] contains 23 categories of synthetic images transformed from original ImageNet images using textual inversion. These categories include particular style, co-occurrence, weather, time of day, etc. For instance, one category, “at dusk,” converts ImageNet images with the prompt “a photo of a [inverse image token] at dusk”. We generated differences analogously to ImageNet-R, taking $\mathcal{D}_A$ to be 500 image samples from the category and $\mathcal{D}_B$ to be 500 original ImageNet images.

PairedImageSets: ImageNetR and ImageNet* mainly capture stylistic differences, and only contain 37 differences in total. To address these shortcomings, we construct *PairedImageSets*, consisting of 150 paired image sets representing diverse differences. The dataset was built by first prompting GPT-4 to generate 150 paired sentences with three difficulty levels of differences (see section A.1 for exact prompts). Easy level represents apparent difference (e.g., “dogs playing in a park” vs. “cats playing in a park”), medium level represents fine-grained difference (e.g., “SUVs on the road” vs. “sedans on the road”), and hard level represents subtle difference (e.g., “people practicing yoga in a mountainous setting” vs. “people meditating in a mountainous setting”).

Once GPT-4 generates the 150 paired sentences, we manually adjusted the annotated difficulty levels to match the criteria above. We then retrieved the top 100 images from Bing for each sentence. As a result, we collected 50 easy, 50 medium, and 50 hard paired image sets, with 100 images for each set. One example pair from this dataset is shown in Figure 2.3, with further examples and a complete list of paired sentences provided in section A.1. We

Proposer	Ranker	ImageNet-R/*		PIS-Easy		PIS-Medium		PIS-Hard	
		Acc@1	Acc@5	Acc@1	Acc@5	Acc@1	Acc@5	Acc@1	Acc@5
Feature (BLIP-2)	Feature (CLIP)	0.68	0.85	0.48	0.69	0.13	0.33	0.12	0.23
Image (LLaVA-1.5)	Feature (CLIP)	0.27	0.39	0.71	0.81	0.39	0.49	0.28	0.43
Caption (BLIP-2 + GPT-4)	Caption (Vicuna-1.5)	0.42	0.70	0.60	0.92	0.49	0.77	0.31	0.61
Caption (BLIP-2 + GPT-4)	Image (LLaVA-1.5)	0.78	0.88	0.78	0.99	0.58	0.80	0.38	0.62
Image (GPT-4V)	Feature (CLIP)	0.86	0.92	0.95	1.00	0.75	0.87	0.57	0.74
Caption (BLIP-2 + GPT-4)	Feature (CLIP)	0.78	0.96	0.88	0.99	0.75	0.86	0.61	0.80

Table 2.2: **Results on VisDiffBench.** GPT-4V image-based and BLIP-2 caption-based proposers with CLIP feature-based rankers consistently outperform other proposers and rankers by a large margin. We use the caption-based proposer with the CLIP ranker as the final VisDiff algorithm because it obtains the highest accuracy on the PairedImageSets-Hard and is cheaper than the GPT-4V image proposer.

will release this dataset and the data collection pipeline.

Evaluation

To evaluate performance on VisDiffBench, we ask algorithms to output a description y for each $(\mathcal{D}_A, \mathcal{D}_B)$ pair and compare it to the ground truth y^* . To automatically compute whether the proposed difference is semantically similar to the ground truth, we prompt GPT-4 to categorize similarity into three levels: 0 (no match), 0.5 (partially match), and 1 (perfect match); see [section A.1](#) for the exact prompt.

To validate this metric, we sampled 200 proposed differences on PairedImageSets and computed the correlation of GPT-4’s scores with the average score across four independent annotators. We observe a high Pearson correlation of 0.80, consistent with prior findings that large language models can align well with human evaluations [66, 294].

We will evaluate systems that output ranked lists of proposals for each $(\mathcal{D}_A, \mathcal{D}_B)$ pair. For these systems, we measure Acc@k, which is the highest score of any of the top-k proposals, averaged across all 187 paired image sets.

2.4 Our Method: VisDiff

It is challenging to train a neural network to directly predict y based on $\mathcal{D}_A$ and $\mathcal{D}_B$: $\mathcal{D}_A$ and $\mathcal{D}_B$ can be very large in practice, while currently no model can encode large sets of images and reliably reason over them. Therefore, we employ a two-stage framework for set difference captioning, using a proposer and a ranker [296, 298]. The *proposer* takes random subsets $\mathcal{S}_A \subseteq \mathcal{D}_A$ and $\mathcal{S}_B \subseteq \mathcal{D}_B$ and proposes differences. The *ranker* takes these proposed differences and evaluates them across all of $\mathcal{D}_A$ and $\mathcal{D}_B$ to assess which ones are most true. We explore different choices of the proposer and ranker in the next two subsections. Full experiment details for this section, including the prompts for the models, can be found in [section A.2](#).

Proposer

The proposer takes two subsets of images $\mathcal{S}_A$ and $\mathcal{S}_B$ as inputs and outputs a list $\mathcal{Y}$ of natural language descriptions that are (ideally) more true on $\mathcal{S}_A$ than $\mathcal{S}_B$. We leverage visual language models (VLM) as the proposer in three different ways: from the images directly, from the embeddings of the images, or by first captioning images and then using a language model. In all cases, we set $|\mathcal{S}_A| = |\mathcal{S}_B| = 20$.

Image-based Proposer: We arrange the 20+20 input images into a single 4-row, 10-column grid and feed this as a single image into a VLM (in our case, LLaVA-1.5 [160] and GPT-4V [193]). We then prompt the VLM to propose differences between the top and bottom half of images.

Feature-based Proposer: We embed images from $\mathcal{S}_A$ and $\mathcal{S}_B$ into the VLM’s visual representation space, then subtract the mean embeddings of $\mathcal{S}_A$ and $\mathcal{S}_B$. This subtracted embedding is fed into VLM’s language model to generate a natural language description of the difference. We use BLIP-2 [148] for this proposer.

Caption-based Proposer: We first use the VLM to generate captions of each image in $\mathcal{S}_A$ and $\mathcal{S}_B$. Then, we prompt a pure language model to generate proposed differences between the two sets of captions. We use BLIP-2 to generate the captions and GPT-4 to propose differences.

Experiments in section 2.5 show that the caption-based proposer works best, so we will take it as our main method and the other two as baselines. To further improve performance, we run the proposer multiple times over different sampled sets $\mathcal{S}_A$ and $\mathcal{S}_B$, then take the union of the proposed differences as inputs to the ranker.

Ranker

Since the proposer operates on small subsets $\mathcal{S}_A$ and $\mathcal{S}_B$ and could generate invalid or noisy differences, we employ a *ranker* to validate and rank the proposed differences $y \in \mathcal{Y}$. The ranker sorts hypotheses by computing a difference score $s_y = \mathbb{E}_{x \in \mathcal{D}_A} v(x, y) - \mathbb{E}_{x \in \mathcal{D}_B} v(x, y)$, where $v(x, y)$ is some measure of how well the image x satisfies the hypothesis y . As before, we leverage VLMs to compute the ranking score $v(x, y)$ in three ways: from images directly, from image embeddings, and from image captions.

Image-based Ranker: We query the VQA model LLaVA-1.5 [160] to ask whether the image x contains y , and set $v(x, y) = \text{VQA}(x, y)$ to be the resulting binary output.

Caption-based Ranker: We generate a caption c from x using BLIP-2 [148], then ask Vicuna-1.5 [39] whether the caption c contains y . We set $v(x, y) = \text{QA}(c, y)$ to be the resulting binary output.

Feature-based Ranker: We use CLIP ViT-G/14 [213] to compute the cosine similarity between the image embedding e_x and text embedding e_y , so that $v(x, y) = \frac{e_x \cdot e_y}{\|e_x\| \|e_y\|}$. In contrast to the other two scores, since $v(x, y)$ is continuous rather than binary, we compute s_y as the AUROC of using v to classify between $\mathcal{D}_A$ and $\mathcal{D}_B$.

Experiments in [section 2.5](#) show that the feature-based ranker achieves the best performance and efficiency, so we use it as our main method and the other two as baselines. We also filter out proposed differences that are not statistically significant, by running a t-test on the two score distributions $v(x, y)$ with significance threshold 0.05.

2.5 Results

In this section, we present experimental results to understand 1) which proposer / ranker works best, 2) can our algorithm consistently find the ground truth difference, and 3) can our algorithm work under noisy settings.

Which Proposer is Best?

Our comparative results, presented in [Table 2.2](#), demonstrate that *the caption-based proposer consistently outperforms its image-based and feature-based counterparts by a large margin* across all subsets of the VisDiffBench. This difference is particularly pronounced in the most challenging subset, PairedImageSets-Hard. While the captioning process may result in some loss of information from the original images, the strong reasoning capabilities of large language models effectively compensate for this by identifying *diverse* and *nuanced* differences between image sets. We provide a qualitative example in [Figure 2.3](#).

The LLaVA image-based proposer shows commendable performance on PairedImageSets-Easy but significantly lags behind the caption-based proposer on the PairedImageSets-Medium/Hard subsets. Similarly, GPT-4V outperforms the caption-based proposer on the easy subset but underperforms on the hard subset. This discrepancy can be attributed to the loss of visual details when aggregating numerous images into a single gridded super-image.

The feature-based proposer outperforms the LLaVA image-based proposer on ImageNetR and ImageNet* but is much less effective across all subsets of PairedImageSets. We believe this is because the feature-based approach excels at distinguishing groups when one possesses attributes absent in the other (e.g., “clipart of an image” minus “an image” equates to “clipart”). Most cases in ImageNetR/ImageNet* fit this scenario. However, this approach falls short in other situations where vector arithmetic does not yield meaningful semantic differences (e.g., “cat” minus “dog” is not semantically meaningful), which is a common scenario in PairedImageSets.

Which Ranker is Best?

In [Table 2.2](#), our results demonstrate that *the feature-based ranker consistently outperforms both the caption-based and image-based rankers*, particularly in the most challenging subset, PairedImageSets-Hard. The feature-based approach’s advantage is primarily due to its continuous scoring mechanism, which contrasts with the binary scores output by image-based and caption-based question answering. This continuous scoring allows for more fine-grained image annotation and improved calibration. It is also logical to observe the image-based

ranker outperforms the caption-based one, as answering questions from original images tends to be more precise than from image captions.

Moreover, the efficiency of the feature-based ranker is remarkable. In scenarios where M hypotheses are evaluated on N images with $N \gg M$, the computation of image features is required only once. This results in a computational complexity of $O(M + N) \approx O(N)$, compared to $O(MN)$ for both image-based and caption-based rankers. Hence, the feature-based ranker requires significantly less computation, especially when ranking many hypotheses. This efficiency is crucial in practical applications, as we have found that a higher volume of proposed differences is essential for accurately identifying correct differences in the [section A.3](#).

Can Algorithm Find True Difference?

In [Table 2.2](#), the results demonstrate the effectiveness of our algorithm in discerning differences. The best algorithm, comprising a GPT-4 [\[193\]](#) caption-based proposer and a CLIP [\[213\]](#) feature-based ranker, achieves accuracies of 88%, 75%, and 61% for Acc@1, and 99%, 86%, and 80% for Acc@5 on the PairedImageData-Easy/Medium/Hard subsets, respectively. The PairedImageData-Hard subset poses a significant challenge, requiring models to possess strong reasoning abilities to perceive extremely subtle variations, such as distinguishing between “Fresh sushi with salmon topping” and “Fresh sushi with tuna topping”, or possess enough world knowledge to discern “Men wearing Rolex watches” from “Men wearing Omega watches”. Despite these complexities, our model demonstrates impressive performance, accurately identifying specifics like “Sushi with salmon” and “Men wearing Rolex watches”.

Performance Under Noisy Data Splits

In the VisDiffBench dataset, image sets are composed with perfect purity. For instance, $\mathcal{D}_A$ exclusively contains cat images (100%), while $\mathcal{D}_B$ is entirely made up of dog images (100%). However, this level of purity is rare in real-world scenarios. Typically, such sets include a mix of elements – for example, $\mathcal{D}_A$ might comprise 70% cat images and 30% dog images, and $\mathcal{D}_B$ vice versa. To evaluate the robustness of the VisDiff algorithm against such noise, we introduced randomness in VisDiffBench by swapping a certain percentage of images between $\mathcal{D}_A$ and $\mathcal{D}_B$. Here, 0% purity signifies 50% image swapping and an equal distribution of two sets, whereas 100% purity indicates no image swapping.

[Figure 2.4](#) presents the Acc@1 and Acc@5 performance of VisDiff across various purity levels, tested on 50 paired sets within PairedImageSets-Hard. As anticipated, a decline in purity correlates with a drop in accuracy since identifying the difference becomes harder. However, even at 40% purity, Acc@1 remains at 49%, only modestly reduced from 63% at 100% purity. This result underscores the robustness of the VisDiff algorithm to noisy data. It is also worth noting that VisDiff reaches near 0% accuracy at 0% purity, which is

expected since the two sets have exactly the same distribution and our method filters out invalid differences.

Other ablations of VisDiff algorithm. In [section A.3](#), we further discuss how caption style, language model, sample size, and # sampling rounds affect VisDiff performance.

2.6 Applications

We apply the best configuration of our VisDiff method to a set of five applications in computer vision: 1) comparing ImageNet and ImageNetV2 ([section 2.6](#)), 2) interpreting the differences between two classifiers at the datapoint level ([section 2.6](#)), 3) analyzing model errors ([section 2.6](#)), 4) understanding the distributional output differences between StableDiffusionV1 and V2 ([section 2.6](#)), and 5) discovering what makes an image memorable ([section 2.6](#)). Since VisDiff is automatic, we used it to discover differences between (1) large sets of images or (2) many sets of images, thus mass-producing human-interpretable insights across these applications. In this section, we report VisDiff-generated insights including some that can be confirmed with existing work and others that may motivate future investigation in the community. Additional details for each application can be found in [section A.4](#).

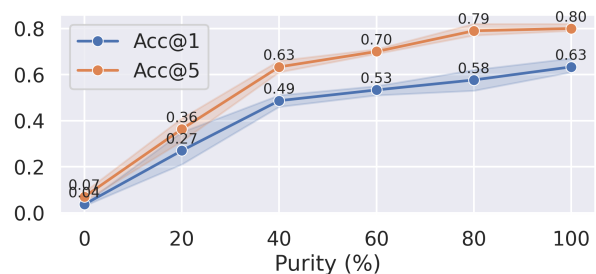


Figure 2.4: **VisDiff performance under noise.** We randomly swap different percentages of images between $\mathcal{D}_A$ and $\mathcal{D}_B$ to inject noise. Results are computed on 50 paired sets in PairedImageSets-Hard. 95% confidence intervals are reported over three runs.

Comparing ImageNetV2 with ImageNet

In 2019, a decade after ImageNet [55] was collected, Recht et al. introduced ImageNetV2 [216], which attempted to mirror the original ImageNet collection process, including restricting data to images uploaded in a similar timeframe. However, models trained on ImageNet showed a consistent 11-14% accuracy drop on ImageNetV2, and the reasons for this have remained unclear. While some studies have employed statistical tools to reveal a distributional difference between ImageNet and ImageNetV2 [74], we aim to discover more interpretable differences between these two datasets.

To uncover their differences, we first ran VisDiff with $\mathcal{D}_A$ as all of ImageNetV2 images and $\mathcal{D}_B$ as all of ImageNet images. Interestingly, the highest scoring description generated by our system is “photos taken from Instagram”. We conjecture that this highlights temporal distribution shift as a potential reason behind model performance drops on ImageNetV2 vs

Class	More True for ImageNetV2
Dining Table	People posing for a picture
Wig	Close up views of dolls
Hand-held Computer	Apps like Twitter and Whatsapp
Palace	East Asian architecture
Pier	Body of water at night

Table 2.3: **Top per-class differences between ImageNet and V2.**

Class	Acc _C	Acc _R	More Correct for CLIP
Tobacco Shop	0.96	0.50	Sign hanging from side
Digital Watch	0.88	0.52	Watches in a group
Missile	0.78	0.42	People posing with missiles
Pot Pie	0.98	0.66	Food size vs. coins
Toyshop	0.92	0.60	People shopping in store

Table 2.4: **Top per-class differences between CLIP and ResNet.** Acc_C/Acc_R: CLIP/ResNet accuracy.

V1. Indeed, while ImageNetV2 aimed to curate images uploaded in a similar timeframe to ImageNet, all images in ImageNet were collected prior to 2012 whereas a portion of ImageNetV2 was collected between 2012 and 2014 [216]. This shift in years happens to coincide with the explosion of social media platforms such as Instagram, which grew from 50M users in 2012 to 300M users in 2014 [58]. In this case, we hypothesize that a small difference in the time range had a potentially outsized impact on the prevalence of Instagram-style photos in ImageNetV2 and the performance of models on this dataset.

Beyond dataset-level analysis, we applied VisDiff to each of the 1,000 ImageNet classes, comparing ImageNetV2 images ($\mathcal{D}_A$) against ImageNet images ($\mathcal{D}_B$). Notable class-specific differences are listed in Table 2.3, ranked by difference score, with visualizations in Figure A.6. Several of these differences suggest more specific examples of Instagram-style photos, consistent with our dataset-level finding. For example, for the class “Dining Table”, ImageNetV2 contains substantially more images showing “people posing for a picture”, visualized in Figure 2.1. For the class “Horizontal Bar”, ImageNetV2 is also identified to have more images of “men’s gymnastics events.” Upon manual inspection, we find that this highlights the difference that ImageNetV2 happens to contain photographs of the Men’s High Bar gymnastics event in the 2012 Olympics, which occurred after the ImageNet collection date. These examples illustrate how VisDiff can be used as a tool for surfacing salient differences between datasets.

Comparing Behaviors of CLIP and ResNet

In 2021, OpenAI’s CLIP [213] showcased impressive zero-shot object recognition, matching the fully supervised ResNet [100] in ImageNet accuracy while showing a smaller performance drop on ImageNetV2. Despite similar in-distribution performance on ImageNet, CLIP and ResNet differ in robustness [186]. This naturally leads to two questions: 1) do these models make similar predictions on individual datapoints in ImageNet? 2) on what datapoints does CLIP perform better than ResNet in ImageNetV2?

To investigate these questions, we analyzed ResNet-50 and zero-shot CLIP ViT-H, which achieve similar accuracies of 75% and 72% on ImageNet, respectively. To study the first question, VisDiff was applied to the top 100 classes where CLIP surpasses ResNet. $\mathcal{D}_A$

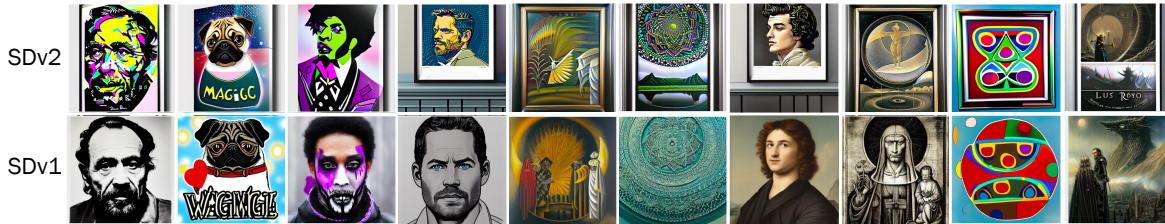


Figure 2.5: **StableDiffusionV2 vs. V1 generated images.** For the same prompt, StableDiffusionV2 images often contain more “vibrant contrasting colors” and “artworks placed on stands or in frames”. Randomly sampled images can be found in [Figure A.9](#).

comprised images correctly identified by CLIP but not by ResNet, and $\mathcal{D}_B$ included all other images. The top discoveries included “close-ups of everyday objects”, “brands and specific product labeling”, and “people interacting with objects”. The first two align well with existing works that show CLIP is robust to object angles and sensitive to textual elements (e.g., a fruit apple with text “iPod” on it will be misclassified as “iPod”) [86, 213]. In addition, we ran VisDiff at finer granularity on each of the top 5 classes where CLIP outperforms ResNet. The discovered class-level differences are shown in [Table 2.4](#), demonstrating CLIP’s proficiency in identifying “tobacco shops with signs”, “group displays of digital watches”, and “scenes involving missiles and toyshops with human interactions”, which echos the dataset-level findings about label, object angle, and presence of people.

To study the second question, we applied VisDiff to ImageNetV2’s top 100 classes where CLIP outperforms ResNet. We set $\mathcal{D}_A$ as images where CLIP is correct and ResNet is wrong, and $\mathcal{D}_B$ as the rest. The top three differences are: 1) “Interaction between humans and objects”, suggesting CLIP’s robustness in classifying images with human presence; 2) “Daytime outdoor environments”, indicating CLIP’s temporal robustness; and 3) “Group gatherings or social interactions”, which is similar to the first difference. These findings provide insight into CLIP’s strengths versus ResNet on ImageNetV2, and are also consistent with the findings in [section 2.6](#) that ImageNetV2 contains more social media images with more presence of people.

Finding Failure Modes of ResNet

We utilize VisDiff to identify failure modes of a model by contrasting images that are correctly predicted against those that are erroneously classified. Using a ResNet-50 and ResNet-101 [100] trained on ImageNet, we set $\mathcal{D}_A$ as ImageNet images misclassified by both ResNet-50 and ResNet-101 and $\mathcal{D}_B$ as correctly classified images. The two highest scoring descriptions were “humanized object items” and “people interacting with objects”, suggesting that ResNet models perform worse when the images include human subjects, echoing the finding in [section 2.6](#).

To validate this hypothesis, we applied a DETR [217] object detector to find a subset of ImageNet images with human presence. Using the classes which have a roughly equal number

	ResNet50	ResNet101	Ensemble
Images w/o Person	69.96%	72.30%	77.32%
Images w/ Person	67.24% $\downarrow 2.72\%$	68.75% $\downarrow 3.55\%$	74.86% $\downarrow 2.46\%$

Table 2.5: **Accuracy on images with / without people.**

of human/no-human images, we evaluated ResNets on this subset and their accuracy indeed declined 3-4%, as shown in [Table 2.5](#).

Comparing Versions of Stable Diffusion

In 2022, Stability AI released StableDiffusionV1 (SDv1), followed by StableDiffusionV2 (SDv2) [223]. While SDv2 can be seen as an update to SDv1, it raises the question: What are the differences in the images produced by these two models?

Using the prompts from PartiPrompts [281] and DiffusionDB [274], we generated 1634 and 10000 images with SDv2 and SDv1, respectively. The Parti images are used to propose differences and the DiffusionDB images are used to validate these differences transfer to unseen prompts.

The top differences show that SDv2 produces more “vibrant and contrasting colors” and interestingly “images with frames or borders” (see [Table A.5](#)). We confirmed the color difference quantitatively by computing the average saturation: 112.61 for SDv2 versus 110.45 for SDv1 from PartiPrompts, and 97.96 versus 93.49 on unseen DiffusionDB images. Qualitatively, as shown in [Section Figure 2.5](#), SDv2 frequently produces images with white borders or frames, a previously unknown characteristic. This is further substantiated in [Section section A.4](#), where we employ edge detection to quantify white borders, providing 50 random image samples from both SDv1 and SDv2.

Describing Memorability in Images

Finally, we demonstrate the applicability of VisDiff in addressing diverse real-world questions beyond machine learning, such as computational cognitive science. A key area of interest, especially for photographers and advertisers, is enhancing image memorability. Isola et al. [115] explored this question and created the LaMem dataset, where each image is assigned a memorability score by humans in the task of identifying repeated images in a sequence.

Applying VisDiff to the LaMem dataset, we divided images into two groups: $\mathcal{D}_A$ (the most memorable 25th percentile) and $\mathcal{D}_B$ (the least memorable 25th percentile). Our analysis found that memorable images often include “presence of humans”, “close-up views”, and “humorous settings”, while forgettable ones feature “landscapes” and “urban environments”. These findings are consistent with those of Isola et al. [115], as further detailed qualitatively in [Figure 2.6](#) and quantitatively in [section A.4](#).



Figure 2.6: **Memorable(top) vs. forgettable(bottom) images.** Memorable images contain more “humans”, “close-up views of body part or objects”, and “humorous settings”, while forgettable images contain more “landscapes” and “urban environments”

2.7 Conclusion

In this work, we introduce the task of set difference captioning and develop VisDiff, an algorithm designed to identify and describe differences in image sets in natural language. VisDiff first uses captioning and large language models to propose differences based on image captions and then employs CLIP to effectively rank these differences. We evaluate VisDiff’s various design choices on our curated VisDiffBench, and show VisDiff’s utility in finding interesting insights across a variety of real-world applications.

Limitations. As we see in Section 2.5, VisDiff still has a large room for improvement and hence far from guaranteed to uncover all meaningful differences. Furthermore, VisDiff is meant to be an assistive tool for humans to better understand their data and should not be applied without a human in the loop: the users hold the ultimate responsibility to interpret the descriptions by VisDiff properly. As VisDiff relies heavily on CLIP, GPT, and BLIP, any biases or errors these models may extend to VisDiff. Further investigation of VisDiff’s failure cases can be found in [section A.5](#).

Chapter 3

Discovering Divergent Representations between Text-to-Image Models

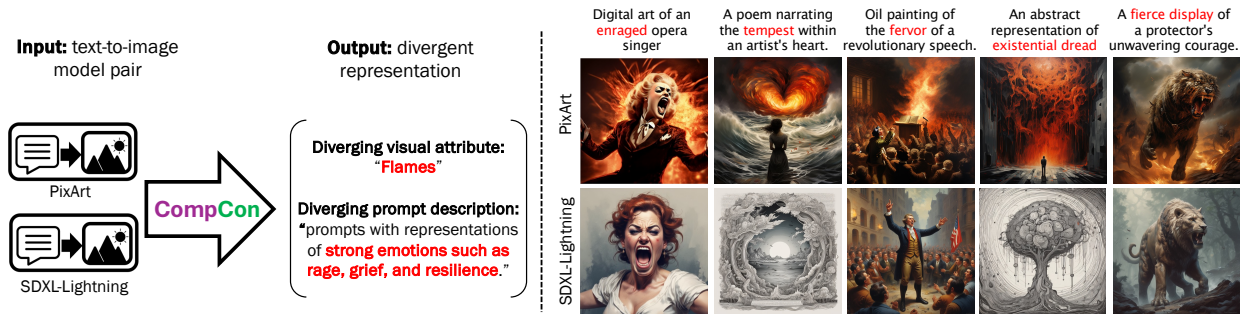


Figure 3.1: **CompCon overview.** CompCon discovers pairs of prompt concepts and visual attributes where two text-to-image models diverge in their outputs.

3.1 Introduction

Generative models develop unique representations of semantic concepts – for instance, happy scenes contain warm colors, or dogs are found outside. While many of these representations are shared across models, understanding when representations **diverge** can reveal stylistic differences between models. In this work, we explore how to uncover such divergent representations by identifying input-dependent differences between two text-to-image models. Specifically, we aim to discover pairs of semantic concepts and visual attributes where prompts containing a semantic concept cause one model to generate images displaying the corresponding visual attribute, while the other does not. For example, prompts that mention strong emotions result in images with flames in one model but not the other (Fig. 3.1).

Discovering divergent representations is beneficial for both model developers and users. For developers, it can help decide which model to deploy to production based on any problematic discovered divergent representations, and for evaluating against competitor models. For users, it can help in selecting the model that best aligns with their own interpretations and needs. Manually performing this task is labor-intensive, as it requires sifting through hundreds or thousands of images to find visual attribute differences; once identified, additional effort is needed to determine the types of input prompts that trigger these differences.

When comparing text-to-image models, the evaluation typically focuses on metrics such as image quality and prompt adherence [91, 108, 109, 111, 144, 204, 228]. While these metrics indicate *how well* models perform, they often overlook *what* the models actually learn. For example, what defines ‘cute’ versus ‘ugly’? What characteristics make something appear ‘futuristic’? What does ‘emotion’ look like? As we will show, models trained on different data, using different encoders or training procedures, can learn distinct interpretations of the same concept. For instance, one model may associate ‘ancient’ with the Paleolithic Era while the other associates it with the Roman Empire.

To address the task of discovering divergent representations, we make the following contributions. First, we introduce CompCon (Comparing Concepts), an evolutionary search algorithm designed to uncover input-dependent differences in model representations. CompCon first discovers pairwise differences in model outputs, generates a description of the prompts that cause this difference, and iteratively refines this description by analyzing existing prompts and generating new ones likely to highlight these differences. As shown in Figure 3.1, CompCon can generate prompts that result in model behaviors like putting flames behind opera singers for one model but not the other.

Second, we create ID² (Input-Dependent Differences), a dataset of 60 semantic-visual representations to evaluate the efficacy of our method. Using this dataset, we compare CompCon to LLM, TF-IDF, and VisDiff [70] baselines. As our third contribution, we apply CompCon to compare the PixArt and SD-Lightning text-to-image models, finding, for example, that prompts mentioning anger result in depictions of ‘flames’ in PixArt, prompts mentioning sadness and solitude result in ‘wet streets’ in PixArt, and abstract prompts with cosmic motifs result in ‘mandala circular designs’ in SD-Lightning. We also uncover bias, such as PixArt generating older men for prompts mentioning traditional professions. These findings demonstrate how CompCon can systematically reveal subtle differences between generative models, helping developers and users better understand and leverage these models’ unique behaviors.

3.2 Related Work

Evaluating Text-to-Image Models. The evaluation of text-to-image models has advanced significantly in recent years. Traditional quality measures such as FID [109], Inception core [228], CLIP score [108], and CLIP-R score [204] have been complemented by newer metrics like TAIM [91] and TIFA [111], which leverage vision-and-language models to

assess prompt adherence. Holistic benchmarks such as HEIM and others [45, 144, 225, 282] aggregate multiple axes of evaluation, including quality, prompt adherence, style, and efficiency. While these approaches excel in measuring overall model performance, they focus on objective qualities and often overlook fine-grained, subjective differences between models. Our work complements these efforts by targeting input-dependent differences, particularly in semantic interpretations and stylistic variations, which are crucial for understanding model-specific behaviors.

Interpreting Diffusion Models. Several works have explored ways of interpreting the internal representations of generative vision-language models. Bau et al. [16], Dravid et al. [63], Gandelsman et al. [79, 80] explore how to describe the function of certain neurons and attention heads in natural language, and Tong et al. [255] discovers how image and text representations differ in latent space to better understand CLIP failures. We see our work as a data-driven approach to attain similar insights into model behavior.

Discovering Bias in Diffusion Models. Uncovering and mitigating biases in text-to-image models has been well explored, with many works focus on finding and mitigating a predefined set of biases related to gender, race, and geography [19, 44, 77, 85, 98, 269, 273]. Recently, a line of works have emerged that aim to automatically discover biases from the data, rather than using a predefined list. Many of these approaches [59, 60] discover bias by prompting a large language model (LLM) to propose potential biases from image captions, generating prompts that may indicate a bias in models (*e.g.*, “a doctor”) and using a VLM to check if this bias exists. Liu et al. [163] builds on this by clustering generated images based on concepts like gender, while Chinchure et al. [41] extends bias discovery to counterfactual examples, eliminating the need for a large caption pool. In contrast, our work focuses on model *comparison*, rather than single-model auditing, which better reflects many real-world evaluations where success is measured by improvement relative to other models. Additionally, while previous methods [59, 60] identify biases from input captions, CompCon analyzes generated images directly using VLMs. This enables discovery of subtler, more nuanced differences in visual representation, including social biases and stylistic or conceptual divergences between models. Further discussion is in Sections 3.5 and A.12.

Describing Differences in Image Sets. Several works have aimed to describe differences in sets of images using natural language. For example, VisDiff [70] generates visual attributes distinguishing two sets of images by analyzing captions and refining them using cross-modal embeddings. Similarly, Chiquier et al. [42] train interpretable CLIP classifiers that evolve based on LLM-generated attributes. While these approaches focus on dataset-level differences, they do not address input-dependent variation. We adapt and extend VisDiff’s methodology for pairwise comparison, focusing on prompt-specific divergences and refining attribute discovery to capture subtler differences. Additionally, our method introduces a novel iterative search for prompt descriptions that cause these differences.

Discovering Divergent Visual Attributes



Discovering Divergent Prompt Descriptions

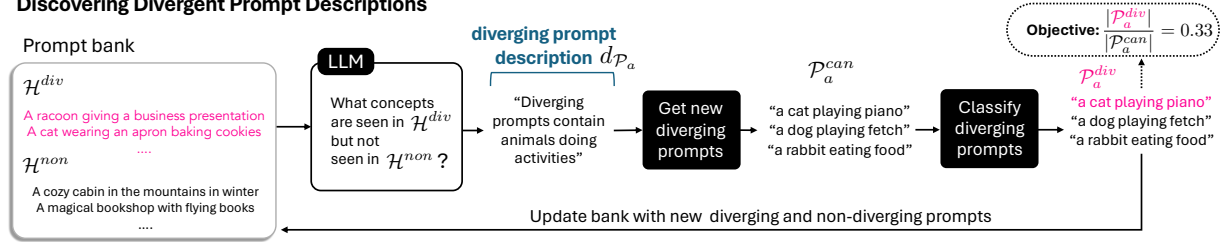


Figure 3.2: **CompCon overview.** We illustrate our approach for discovering diverging visual attributes (top) and diverging prompt descriptions (bottom). Given two text-to-image models and a set of prompts, we use a VLM to identify visual differences. For each diverging attribute, we iteratively refine diverging prompt descriptions by generating candidate prompts $\mathcal{P}_a^{can}$ from the description, classifying them as diverging ($\mathcal{H}^{div}$) or non-diverging ($\mathcal{H}^{non}$). The objective is to maximize the proportion of generated prompts classified as diverging.

3.3 Divergent Representation Discovery

Let P be a set of text prompts, and $\mathcal{I}_1^{(P)}$ and $\mathcal{I}_2^{(P)}$ be sets of images generated by two text-to-image models given the input prompts P . We call the natural language descriptions of the prompt set P and any visual differences between the two generated image sets $\mathcal{I}_1^{(P)}$ and $\mathcal{I}_2^{(P)}$ a *divergent representation* (see Figure 3.1). These divergent representations take the form of a pair of natural language descriptions (a, d_{P_a}) , where a is a description of a visual attribute seen more often in one model than the other (e.g., “flames”), and d_{P_a} is a description of the concepts present in text prompts P_a eliciting this difference (e.g., “strong emotions”). We refer to a as a *diverging visual attribute*, d_{P_a} as a *diverging prompt description*, and P_a as *diverging prompts*.

Given a pair of text-to-image models $\Theta = (\theta_1, \theta_2)$, we aim to discover differences in the images generated by the two models as well as a description of the types of input text prompts that trigger these differences. Let $\mathcal{P}$ be the set of all possible text prompts and $\mathcal{A}$ be the set of all possible diverging visual attributes. Our goal is to find the mapping $\mathcal{F}_\Theta$ from text prompts in $\mathcal{P}$ to diverging visual attributes in $\mathcal{A}$ given the model pair Θ ,

$$\mathcal{F}_\Theta : \mathcal{P} \mapsto \mathcal{A}. \quad (3.1)$$

Note that this mapping is not a bijection as multiple text prompts may map to a single diverging visual attribute. Moreover, multiple diverging visual attributes may be depicted for a given set of diverging prompts.

Our approach for computing the mapping $\mathcal{F}_\Theta$ comprises two steps, illustrated in Figure 3.2. First, given the text-to-image model pair Θ and a large set of initial prompts $\mathcal{P}_0 \subset \mathcal{P}$, we compute a set of diverging visual attributes $\mathcal{A}_0 \subset \mathcal{A}$ (Section 3.3). Next, for each diverging visual attribute $a \in \mathcal{A}_0$, we optimize an objective to find the set of diverging prompts $\mathcal{P}_a \subset \mathcal{P}$ resulting in the diverging visual attribute a (Section 3.3). We next describe each of these steps.

Discovering Diverging Visual Attributes

Our goal is, given a text-to-image model pair Θ and a large set of initial text prompts $\mathcal{P}_0$, to compute a set of diverging visual attributes $\mathcal{A}_0$ over images generated given the prompts. This task is challenging as a system must identify any consistent visual differences between the two models' generated image sets. These differences are often subtle and difficult to spot over the large generated image collection. We address this challenge by prompting an off-the-shelf vision-language model (VLM) for this task.

For the text-to-image model pair Θ and a text prompt p , let $\mathcal{G}(\Theta, p) = \{\mathcal{I}_1^{(p)}, \mathcal{I}_2^{(p)}\}$ denote the two sets of images generated by each model given prompt p . We first sample a batch of prompts $\mathcal{P}_{batch} \subset \mathcal{P}_0$ and, for each prompt $p \in \mathcal{P}_{batch}$, we construct a two-row image grid by tiling the images in $\mathcal{I}_1^{(p)}$ on the top row and $\mathcal{I}_2^{(p)}$ on the bottom row. Using this image grid, we instruct a VLM to find diverging visual attributes appearing more in images of $\mathcal{I}_1^{(p)}$ compared to $\mathcal{I}_2^{(p)}$ (see Appendix for our instruction prompt). Our resulting diverging visual attribute list $\mathcal{A}_0$ is the aggregation of discovered attributes across $\mathcal{P}_{batch}$.

Next, we rank each diverging visual attribute $a \in \mathcal{A}_0$ by assigning a score indicating how well attribute a can distinguish image sets $\mathcal{I}_1^{(\mathcal{P}_0)}$ and $\mathcal{I}_2^{(\mathcal{P}_0)}$. For each set of images generated by prompt $p \in \mathcal{P}_0$, we define a *divergence score* $z(a, \mathcal{I}_1^{(p)}, \mathcal{I}_2^{(p)}) \rightarrow \{0, 1\}$ that indicates whether image set $\mathcal{I}_1^{(p)}$ contains attribute a while $\mathcal{I}_2^{(p)}$ does not.

Using cross-modal similarity, here CLIP [211], we compute the cosine similarity $s(\cdot)$ between diverging visual attribute a and each image in sets $\mathcal{I}_1^{(p)}$ and $\mathcal{I}_2^{(p)}$. Using these similarities, we define the divergence score as the product of two indicated conditions,

$$z(a, \mathcal{I}_1^{(p)}, \mathcal{I}_2^{(p)}) = \mathbb{1}[s(a, \mathcal{I}_1^{(p)}) > t] \times \mathbb{1}[s(a, \mathcal{I}_1^{(p)}) - s(a, \mathcal{I}_2^{(p)}) > \delta] \quad (3.2)$$

where t and δ are hyperparameters that determine if $\mathcal{I}_1^{(p)}$ contains attribute a and $\mathcal{I}_2^{(p)}$ does not contain a .

Using this divergence score, we define the overall score for attribute a as the mean divergence score over prompts in the initial prompt set: $\frac{1}{|\mathcal{P}_0|} \sum_{p \in \mathcal{P}_0} z(a, \mathcal{I}_1^{(p)}, \mathcal{I}_2^{(p)})$. A score of 1 means that all images generated by model θ_1 contain attribute a and none of the images

Algorithm 1 Discovering diverging prompt descriptions

Input: Model pair Θ , diverging visual attribute a , initial set of prompts $\mathcal{P}_0$, number of iterations N

Output: Diverging prompt description $d_{\mathcal{P}_a}^*$

```

1: Initialize: prompt bank  $\mathcal{H} \leftarrow \emptyset$ , scores  $\sigma \leftarrow \text{emptyArray}(N)$ , descriptions  $d_{\mathcal{P}_a} \leftarrow \text{emptyArray}(N)$ 
2:  $\mathcal{P}_a \leftarrow \text{classifyDiverging}(\Theta, a, \mathcal{P}_0)$ 
3:  $\mathcal{H}^{div} \leftarrow \mathcal{P}_a$ 
4:  $\mathcal{H}_a^{non} \leftarrow \mathcal{P} \setminus \mathcal{P}_a$ 
5: for  $i$  in  $1, \dots, N$  do
6:    $h_a^{div}, h_a^{non} \leftarrow \text{sample}(\mathcal{H}^{div}, \mathcal{H}^{non})$ 
7:    $d_{\mathcal{P}_a}[i] \leftarrow \text{describeDiverging}(\mathcal{P}_a^{div}, \mathcal{P}_a^{non}, \mathcal{H})$ 
8:    $\mathcal{P}_a^{can} \leftarrow \text{getNewDiverging}(d_{\mathcal{P}_a}[i], a, \mathcal{H})$ 
9:    $\mathcal{P}_a^{div}, \mathcal{P}_a^{non} \leftarrow \text{classifyDiverging}(\Theta, a, \mathcal{P}_a^{can})$ 
10:   $\sigma[i] \leftarrow |\mathcal{P}_a^{div}|/|\mathcal{P}_a^{can}|$ 
11:   $\mathcal{H}^{non} = \mathcal{H}^{non} \cup \mathcal{P}_a^{non}$ 
12:   $\mathcal{H}^{div} = \mathcal{H}^{div} \cup \mathcal{P}_a^{div}$ 
13: end for
14:  $d_{\mathcal{P}_a}^* \leftarrow d_{\mathcal{P}_a}[\text{argmax}(\sigma)]$ 
15: return  $d_{\mathcal{P}_a}^*$ 
    
```

generated by θ_2 contain a . A score of 0 indicates that a never appears more often in images generated by θ_1 . Note that we are not optimizing for scores close to 1, we are simply interested in any attribute a that obtains a score sufficiently above zero. Finally, as many attributes $\mathcal{A}_0$ are semantically equivalent (e.g. "flames" and "fire") we prompt an LLM to remove similar attributes.

Discovering Diverging Prompt Descriptions

After we discover a set of diverging visual attributes $\mathcal{A}_0$, for each attribute $a \in \mathcal{A}_0$ we aim to find a natural language description $d_{\mathcal{P}_a}$ of the diverging prompts that trigger this attribute. This task is challenging as the search space over all possible text prompts $\mathcal{P}$ is large, and we must not only find a set of prompts but a natural language description that completely covers this set.

Let $\mathcal{L}(d_{\mathcal{P}_a})$ be the diverging prompts generated from description $d_{\mathcal{P}_a}$. Our objective is to maximize the expected divergence score over the generated prompts:

$$\max_{d_{\mathcal{P}_a}} \mathbb{E}_{p \sim \mathcal{L}(d_{\mathcal{P}_a})} [z(a, \mathcal{I}_1^{(p)}, \mathcal{I}_2^{(p)})], \quad (3.3)$$

where z is the divergence score defined in Equation (3.2). That is, we want to maximize the number of prompts that have been generated by description $d_{\mathcal{P}_a}$ and confirmed to be diverging. We explore two options for our prompt generator $\mathcal{L}$. The first is to use an LLM to

generate new prompts given $d_{\mathcal{P}_a}$, and the second is to retrieve prompts from a large prompt bank using $d_{\mathcal{P}_a}$.

As the desired diverging prompt description $d_{\mathcal{P}_a}$ is discrete (natural language) and Objective (3.3) is not differentiable, we optimize the objective via evolutionary search. To achieve this goal, for N evolutionary search iterations, we use an LLM to generate description $d_{\mathcal{P}_a}$ given a bank of diverging and non-diverging prompts $\mathcal{H}^{div}$ and $\mathcal{H}^{non}$, generate new prompts and images from $d_{\mathcal{P}_a}$, add these new prompts to $\mathcal{H}^{div}$ and $\mathcal{H}^{non}$, and evolve our description using these updated sets.

We maintain a bank $\mathcal{H} = (\mathcal{H}^{div}, \mathcal{H}^{non})$ of diverging and non-diverging text prompts, where diverging prompts are given by $\mathcal{H}^{div} = \{p \mid z(a, \mathcal{I}_1^{(p)}, \mathcal{I}_2^{(p)}) = 1\}$ and non-diverging prompts are given by $\mathcal{H}^{non} = \{p \mid z(a, \mathcal{I}_1^{(p)}, \mathcal{I}_2^{(p)}) = 0\}$. We mutate the current description $d_{\mathcal{P}_a}$ by prompting an LLM (GPT-4o) to provide a description of what concepts are shared in $\mathcal{H}^{div}$ but not in $\mathcal{H}^{non}$ and score the mutation using Objective (3.3). The mutation prompt is provided in the Appendix.

Algorithm 1 provides pseudocode for our evolutionary search algorithm, with its key functions described below.

Mutation function describeDiverging. Given the bank of diverging and non-diverging text prompts $\mathcal{H}$, we sample B prompts from $\mathcal{H}^{div}$ and $\mathcal{H}^{non}$ and instruct an LLM to output a description $d_{\mathcal{P}_a}$ (the diverging prompt description) of what concepts are shared across diverging prompts which are not seen in non-diverging prompts. After the first iteration, prompts in $\mathcal{H}$ which were generated in the previous iteration are up-weighted when sampling.

Mutation scoring. To score the current mutation, we first define a function `getNewDiverging` that, given the diverging prompt description $d_{\mathcal{P}_a}$, diverging visual attribute a , and prompt bank $\mathcal{H}$, provides candidate prompts $\mathcal{P}_a^{can}$ that are likely to be diverging and do not directly relate to attribute a . We explore two ways of obtaining $\mathcal{P}_a^{can}$: generation and retrieval. In the generation setting, we prompt an LLM (GPT-4o) to generate a diverse set of k new prompts that align with the description $d_{\mathcal{P}_a}$, given random samples of prompts from $\mathcal{H}$ as a point of reference. The instruction prompt can be found in the Appendix. In the retrieval setting we use description $d_{\mathcal{P}_a}$ to retrieve the top k prompts from the prompt bank $\mathcal{H}$ excluding the B sampled prompts used to generate the prompt description, having the highest text embedding similarity to $d_{\mathcal{P}_a}$. We show results of both of these approaches in Section 3.5 and provide further implementation details in 3.5.

Next, we define a function `classifyDiverging` that, given the model pair Θ , visual attribute a , and candidate prompts $\mathcal{P}_a^{can}$, finds the diverging prompts $\mathcal{P}_a^{div} = \{p \in \mathcal{P}_a^{can} \mid z(a, \mathcal{I}_1^{(p)}, \mathcal{I}_2^{(p)}) = 1\}$. If more than one image is generated per prompt, we define a prompt as diverging if the majority of the generated images result in a diverging score $z(a, i_1^{(p)}, i_2^{(p)}) = 1$ for $i_1^{(p)} \in \mathcal{I}_1^{(p)}$ and $i_2^{(p)} \in \mathcal{I}_2^{(p)}$.

Finally, we approximate the expectation in Objective (3.3) by the ratio of diverging to candidate prompts set sizes,

$$\mathbb{E}_{p \sim \mathcal{L}(d_{\mathcal{P}_a})}[z(a, \mathcal{I}_1^{(p)}, \mathcal{I}_2^{(p)})] = \frac{|\mathcal{P}_a^{div}|}{|\mathcal{P}_a^{can}|} \quad (3.4)$$

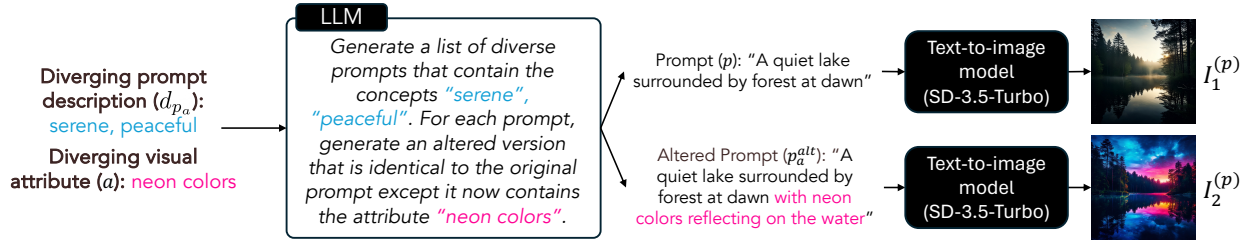


Figure 3.3: **ID² creation.** Given a diverging prompt description d_{p_a} and diverging visual attribute a , we use an LLM to generate prompt pairs where one of the prompts mentions the diverging visual attribute. Both prompts are then passed to the same text-to-image model to generate image pairs with the visual difference a .

where $|\cdot|$ denotes the size of the set. We return the diverging prompt description d_{p_a} with the highest ratio.

3.4 ID² Dataset and LLM Evaluation

To systematically evaluate methods for discovering divergent representations, we created ID² (Input-Dependent Differences), a benchmark dataset containing 60 divergent representations between text-to-image models. Each representation consists of a diverging visual attribute and its corresponding diverging prompt description. Moreover, we include for each diverging representation a set of prompts that align with the diverging prompt description and, for each prompt, pairs of generated images where the diverging visual attribute is depicted in one image but not the other.

Creating such a benchmark is challenging because divergent representations between models are not known *a priori*, and manual annotation across the vast prompt space is impractical. To address this, we use a simulation approach: rather than comparing two distinct models, we use a single model (SD-3.5-Turbo [3]) and simulate differences by modifying input prompts to include specific visual attributes.

For each divergent representation in ID², we generate paired prompts where one explicitly mentions the visual attribute while the other does not. Both prompts are processed by the same text-to-image model, creating image pairs that exhibit controlled, systematic differences. This simulation approach allows us to create ground truth data with known divergent representations against which we can evaluate discovery methods. To assess the quality of discovered representations, we developed an LLM-based evaluation framework that measures both attribute similarity and description accuracy compared to ground truth. This framework provides objective metrics to compare different approaches for identifying divergent representations between text-to-image models. We describe the dataset creation and evaluation process below and in Figure 3.3.

ID² Dataset Creation. Each divergent representation (a, d_{p_a}) in ID² comprises a ground truth diverging visual attribute a and ground truth diverging prompt description d_{p_a} . We

generate the diverging representations using either GPT-4o, Claude 3.5 Sonnet [7], or manually. Each diverging visual attribute a is a short phrase, and diverging prompt description d_{P_a} is a list of semantic concepts that a prompt should include. These representations cover diverse categories, such as related concepts (‘red roses’, [‘love’, ‘romance’]), abstract representations (‘geometric shapes’, [‘efficiency’, ‘productive’]), and bias (‘overweight person’, [‘lazy’, ‘unmotivated’]). The full list of diverging representations are located in the Appendix.

Given a diverging representation (a, d_{P_a}) , we generate a corresponding set of prompts and generated images $\{(p, \mathcal{I}_1^{(p)}, \mathcal{I}_2^{(p)})\}$ such that each prompt p aligns with diverging prompt description d_{P_a} and the accompanying pair of generated images $\mathcal{I}_1^{(p)}$ and $\mathcal{I}_2^{(p)}$ has diverging visual attribute a appearing in one generated image but not the other. We illustrate the process for creating this set in Figure 3.3. First, we use GPT-4o to generate a prompt p that aligns with diverging prompt description d_{P_a} along with an altered prompt p^{alt} that contains attribute a . Both prompts are then given to a diffusion model θ (SD-3.5-Turbo) to generate images such that $\mathcal{I}_1^{(p)} = \theta(p)$ and $\mathcal{I}_2^{(p)} = \theta(p^{\text{alt}})$. The authors manually inspect these generations to ensure the prompts align with the diverging prompt description d_{P_a} , and that diverging visual attribute a is seen in the majority of images corresponding to $\mathcal{I}_1^{(p)}$ but not $\mathcal{I}_2^{(p)}$. We further validate these sets with human studies described later in the section.

For each of the 60 diverging representations, we generate 3 image pairs per prompt across 50 prompts, resulting in two sets of 150 images. Finally, for each representation, we also include a set of distractors – image pairs from 200 randomly generated prompts where $\mathcal{I}_1^{(p)}$ and $\mathcal{I}_2^{(p)}$ are generated with the same prompt p over different random seeds, resulting in image pairs with no discernible difference in visual attributes. The purpose of these distractors is to assess the methods’ ability to generate descriptions and attributes in the presence of noise (no diverging visual attribute present).

LLM Evaluation. When given ground truth representations (a, d_{P_a}) and predicted representations (a', d'_{P_a}) , we use an LLM-as-a-judge (GPT-4o) to compute an *attribute score* and a *description score*, which respectively measure the similarity of the predicted visual attribute and prompt description to the ground truth. The prompt given to the judge can be found in the Appendix.

These scores range from 0 to 1, where 1 indicates perfect agreement, 0.5 indicates partial agreement, and 0 indicates no agreement. Partial agreement means that the predicted attribute or description is related to the ground truth. For example, the ground truth visual attribute is ‘flames’ and the predicted attribute is ‘a red color palette’ would be deemed a partial alignment. To maintain a fair evaluation with other baselines, we prompt our mutation LLM to structure descriptions d_{P_a} as a short list of semantic concepts.

Human Validation of ID² and LLM Evaluation. We conducted a two-stage validation study with 4 PhD students, each annotating 10 randomly sampled sets from ID² (2 annotations per set). Participants identified concepts appearing more frequently in one set versus another and concepts shared across prompts. They then assessed whether our ground truth diverging attribute and diverging prompt description aligned with their descriptions and with the images/prompts. Nearly all participants agreed with the ground truth: partic-

ipants validated that all 40 visual attributes appeared in generated image pairs, and 36/40 provided free-form attributes that matched the ground truth (match validated by the participants). Similarly, for prompt descriptions, 39/40 validated the ground truth, and 34/40 provided free-form attributes that matched the ground truth.

To validate our LLM scoring system, three participants scored 25 predictions using the same rubric given to the LLM judge. The weighted Cohen’s kappa [51] between humans and the LLM was 0.635, comparable to inter-human agreement (0.667), confirming that LLM-as-a-judge scores align closely with human evaluation.

3.5 Experiments

We measure CompCon’s ability to discover diverging visual attributes and diverging prompt descriptions in comparison to baselines on the ID² (Section 3.5) and apply CompCon to compare two popular open source models, PixArt [32] and SD-Lightning [157] to find diverging representations and uncover age and gender bias (Section 3.5)

Experimental Details

For our experiments, we set the size of $\mathcal{P}_{batch}$ to be 50, $t = 0$, and $\delta = 0.05$ for our diverging visual attribute discovery phase. In our diverging prompt description phase, we sample 25 prompts from $\mathcal{H}^{div}$ and $\mathcal{H}^{non}$ and generate $k = 25$ new prompts per iteration. For all experiments, each model generates 3 images per prompt across different seeds. We set $t = 0.2$ and $\delta = 0.05$ for our benchmark comparison. For the model comparison in Section 3.5, we manually inspect prompts labeled as diverging to set thresholds t and δ . Hyperparameters for each discovered diverging representation, as well as all LLM and VLM prompts used for the method and baselines can be found in the Appendix.

We use GPT-4o [197] as the LLM for CompCon’s diverging visual attribute discovery and diverging prompt description discovery phases, as well as for the LLM-Only baseline described below. We use CLIP ViT-bigG-14 [37, 113, 211] trained on Laion2b [232] to classify diverging prompts and attributes and the instructor-xl [244] text embedding model for prompt retrieval. Our dataset generation and experiments were performed on two 80GB NVIDIA A100 GPUs.

Baselines

We compare CompCon to several baselines on the ID², including an end-to-end approach and approaches for the individual diverging visual attribute and diverging prompt description discovery phases.

LLM-only (end-to-end). We select 50 random prompts P_{sample} from each dataset $\mathcal{D}_a$, caption the corresponding generated images and prompt an LLM (GPT-4o) to find any diverging representations.

TF-IDF (diverging visual attribute discovery). We caption the generated images $\mathcal{I}_1^{(\mathcal{P}_{sample})}$ and $\mathcal{I}_2^{(\mathcal{P}_{sample})}$. We then combine all captions produced for $\mathcal{I}_1^{(\mathcal{P}_{sample})}$ and $\mathcal{I}_2^{(\mathcal{P}_{sample})}$ into two separate documents and compute TF-IDF [241], taking the top-5 1-3 word phrases that appear more often in captions of $\mathcal{I}_1^{(\mathcal{P}_{sample})}$ than $\mathcal{I}_2^{(\mathcal{P}_{sample})}$.

VisDiff (diverging visual attribute discovery). We apply the VisDiff algorithm [70] for finding differences in image sets $\mathcal{I}_1^{(\mathcal{P}_{sample})}$ and $\mathcal{I}_2^{(\mathcal{P}_{sample})}$.

TF-IDF (prompt description discovery). Given a diverging visual attribute a , we run `classifyDiverging` on $\mathcal{D}_a$ to obtain the diverging and non-diverging prompts and run TF-IDF to find which phrases appear more often in the diverging prompts compared to the non-diverging prompts. We report results using this method on the diverging visual attribute’s discovered by CompCon.

We use Llava 1.5-7b [159] for image captioning used in the VisDiff and TF-IDF baselines.

Benchmark Results

Attribute Score			Description Score		
Method	Top 1	Top 5	Method	Top 1	Top 5
CompCon	0.60	0.68	CompCon [5-iter]	0.64	0.78
VisDiff	0.47	0.62	CompCon [1-iter]	0.59	0.72
TF-IDF	0.23	0.37	TF-IDF	0.40	0.57
LLM-only	0.08	0.24	LLM-only	0.03	0.28

Table 3.1: **Visual attribute and prompt description scores on ID².** Our approach outperforms all baselines on both diverging visual attribute and diverging prompt description discovery.

We report diverging visual attribute and diverging prompt description discovery results in Table 3.1. CompCon obtains higher attribute and description scores than the baselines. An example output, along with their strengths and weaknesses, are shown in Figure 3.4. We find that:

1. The LLM-only baseline performs poorly compared to the other methods, often outputting diverging representations that are completely unrelated to the ground truth. This result is due to the complexity of the task: the LLM must (1) find differences between each pair of captions, (2) uncover which differences are seen most often, and (3) summarize the prompts that have this difference in the captions.
2. When discovering diverging visual attributes, VisDiff and TF-IDF often fail at identifying more fine-grained differences. This finding is likely due to the reliance on captions lacking fine-grained detail, especially when comparing images depicting similar contexts.

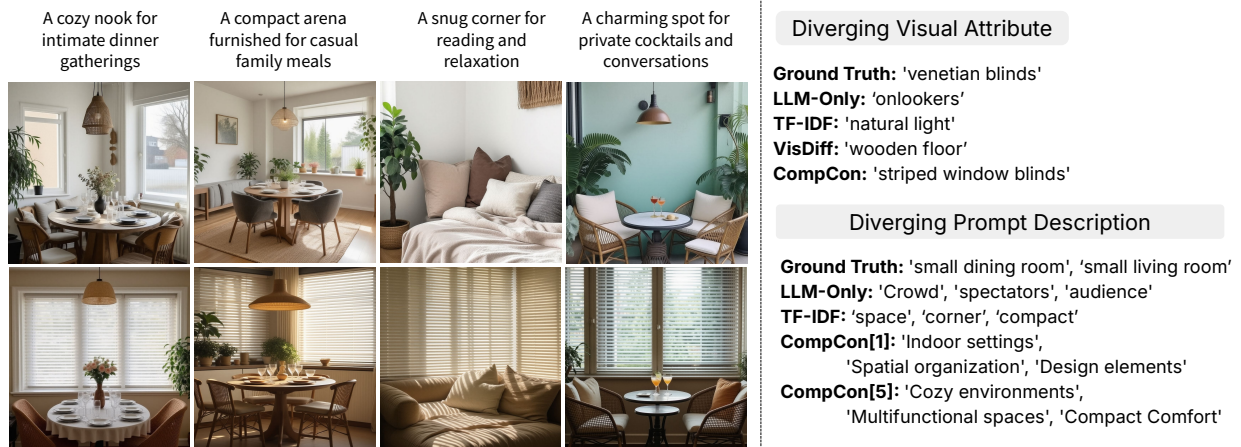


Figure 3.4: **ID² example.** *Top:* We show dataset prompts and corresponding generated images, where the second image row depicts the diverging visual attribute. *Bottom:* We show the ground truth diverging visual attribute and diverging prompt description, along with outputs from our approach and the baselines. Notice that our method produces outputs that better align with the ground truth.

3. While additional iterations offer modest gains in performance when generating diverging prompt descriptions, we see in Figure 3.4 that iterations are beneficial when the ground truth description is more fine-grained. This iterative refinement enables the model to evolve from describing general representations to fine-grained qualities.

Additional experiments on the effects of iterations, the choice of LLM and VLM, and sensitivity analysis of the effects of LLM and VLM errors are in Section A.7.

Qualitative Results

Using CompCon we find divergent representations in several popular diffusion models: PixArt Alpha [32], SDXL-Lightning [157], Stable Diffusion 3.5 Large [3], and Playground 2.5 [147]. As listed in Section 3.3, we use an LLM to generate diverging prompts from the prompt description generated at each iteration. A subset of these generated diverging prompts are shown below. Additional prompts, experiments on other prompt sets and models, and a comparison to a single-model method are in Sections A.6 and A.12.

Results on templated prompts. We run CompCon with an initial prompt bank covering different art styles, subjects, and descriptors and visualize results in Figure 3.5. CompCon discovers both diverging representations like negative emotions and empty urban environments produce “wet streets” in PixArt. Looking at the “flames” example in Figure 3.1, we also see diversity in the presentation of attributes, with flames taking the form of a burning podium to a fiery background to a large red cloud. Lastly, we see scenarios of poor prompt adherence in the “decay” divergence representation, where SD-Lightning does not

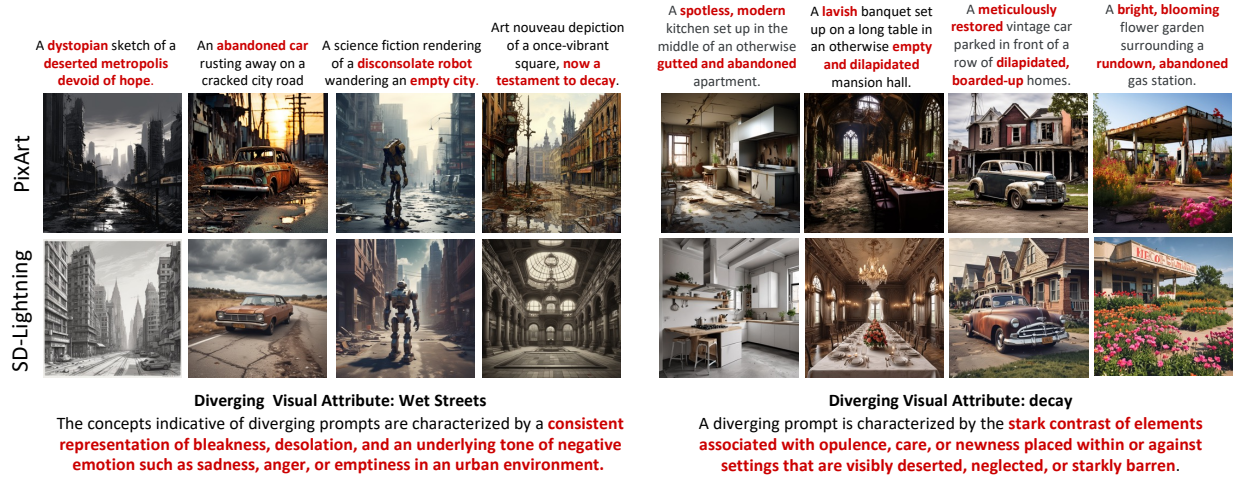


Figure 3.5: **CompCon results comparing PixArt and SD-Lightning.** PixArt associates negative emotions / desolation in urban environments with ‘wet streets,’ while SD-Lightning struggles to depict run-down or dilapidated scenes, where PixArt instead conveys ‘decay.’

produce visual elements that indicate rundown and abandoned places while PixArt associates abandonment with decay. Our findings demonstrate that CompCon can effectively uncover both concrete and abstract divergent representations in text-to-image models, providing interpretable insights into their behavior. The template used to create the prompt bank, additional qualitative examples, and analysis on the effects of iterations are included in the Appendix. Additionally, the Appendix includes results of running CompCon on a dataset of prompts generated by LLMs, creating a fully automated pipeline.

Detecting bias. We show that CompCon can be used for the crucial task of bias detection. As an initial prompt set, we take existing prompts from Luccioni et al. [169], which probe a model’s gender bias when it comes to professions. This dataset contains 252 template prompts that uses a list of professions and interchanges “man”, “woman”, and “person” (*e.g.*, “A man/woman/person who works as a baker”). In Figure 3.6, CompCon highlights differences in how professions are visually represented, with Stable Diffusion 3.5 generating significantly more African American people in media and communication-focused roles. More examples of CompCon detecting other biases are in Section A.6.

Limitations and Cost

As CompCon relies on off-the-shelf LLMs and VLMs, it inherits their biases. While such biases can cause false negatives when discovering diverging representations, the descriptions found still align with human discovery. We validate this through user studies showing that our benchmark evaluation matches human annotation. Additionally, Section A.7 shows that CompCon detects gender and age biases between diffusion models, and CompCon can find correct diverging representations even when the VLM or LLM fails part of the time.



Figure 3.6: **Finding bias.** CompCon discovers both age and racial bias across different text-to-image models. (Left) Age bias in PixArt for prompts about traditional professions. (Right) Racial bias in Stable Diffusion 3.5 for prompts about media professions.

Cost. Using GPT-4o as the VLM and LLM costs $\sim \$0.50$ for attribute discovery plus ~ 0.02 per attribute iteration. This can be cost-effective for comparing one model to N others ($\mathcal{O}(N)$). Running smaller open-source models like IDEFICS llama3-8b [139] cuts costs while maintaining competitive performance, still outperforming baselines. Additional open-model results are in Section A.7.

3.6 Conclusion

We present CompCon, a method for systematically discovering divergent representations between text-to-image models. By identifying input-dependent differences in model outputs and uncovering the prompt concepts linked to these differences, CompCon provides a framework to understand how models interpret semantic concepts differently. Our results on our ID² benchmark and in the comparison of PixArt and SD-Lightning, demonstrate CompCon effectiveness in revealing subtle model-specific behaviors. This opens the possibility of identifying and mitigating unwanted behavior-generated images and videos. Moreover, our approach can serve as a tool for probing the hypothesis that different models converge to the same representation [112].

Chapter 4

From Crowdsourced Data to High-Quality Benchmarks: Arena-Hard and BenchBuilder Pipeline

4.1 Introduction

The proliferation of Large Language Models (LLMs) has spurred advancements, leading to models trained on vast datasets with significantly enhanced capabilities. However, this innovation also presents challenges in designing effective evaluation benchmarks. Many existing benchmarks [21, 23, 62, 103, 214, 267] are unable to keep up with the rapidly advancing capabilities of these models, becoming saturated and unable to differentiate between state-of-the-art models. Additionally, many benchmarks are designed for automatic evaluation with restrictive prompts, such as close-ended or multiple-choice questions, which do not accurately reflect a model’s performance in real-world interactions. Recent developments in benchmark evaluation [247, 294] have made it feasible to evaluate more open-ended prompts, yet generating high-quality prompts at scale remains challenging. Furthermore, static benchmarks are prone to over-fitting and potential test leakage over time, reducing their effectiveness in ranking models.

To address these issues, an effective benchmark must confidently differentiate model capabilities, reflect real-world human interactions, and update frequently to stay relevant and avoid overfitting. Live crowdsourced evaluation platforms, like the Chatbot Arena [40], address some of these concerns by collecting real-world user prompts with preference labels. However, without quality control, these platforms can include low-quality prompts that do not adequately differentiate top models.

In this work, we introduce BenchBuilder, a data pipeline designed to automatically build high-quality benchmarks from live crowdsourced data source such as the Chatbot Arena. BenchBuilder identifies diverse benchmark prompts that can confidently separate model

performance. It uses seven key indicators of high-quality prompts, such as specificity and domain knowledge, and employs an LLM to annotate prompts for these indicators. The resulting benchmark, Arena-Hard-Auto v0.1, uses an LLM judge [293] to estimate human preference against a baseline model, making BenchBuilder and Arena-Hard-Auto v0.1 a fully automated system. We further explore how to measure the quality of benchmarks by introducing metrics to assess a benchmark’s ability to confidently separate models and its agreement with human preferences. Comparing Arena-Hard-Auto v0.1 to current leading benchmarks such as AlpacaEval [247] and MT-Bench [293], we demonstrate that Arena-Hard-Auto v0.1 offers significantly stronger separability with tighter confidence intervals. Additionally, Arena-Hard-Auto v0.1 achieves 89.1% agreement with Chatbot Arena rankings, making it an excellent predictor of downstream performance. Arena-Hard-Auto v0.1 provides a cost-effective and informative evaluation benchmark, and the BenchBuilder pipeline serves as a valuable tool for developers seeking to automatically generate high-quality benchmarks from vast data sources.

To summarize, our work makes the following contributions:

1. We propose a novel data curation pipeline, BenchBuilder, to automatically construct high-quality benchmarks from crowdsourced data.
2. We propose metrics to capture desired properties in a benchmark, and validate Arena-Hard-Auto v0.1 achieves significantly better separability and agreement than existing benchmarks.
3. We open-source both the Arena-Hard-Auto v0.1 benchmark and BenchBuilder pipeline¹.

4.2 Related works

LLM benchmarks. We briefly review widely used LLM benchmarks. Most existing benchmarks are static and ground-truth-based (e.g., multi-choice question answering). They cover a wide range of domains, including math, science, coding, and reasoning. Common ones include MMLU [103], MATH [105], GSM-8K [50], HumanEval [34], DROP [65], BigBench [243], HellaSwag [285], AGIEval [299], GPQA [220], as well as comprehensive collection like HELM [152]. Many have considered task-based evaluation such as IFEval [301], SWE-Bench [124], or AgentBench [165]. As LLMs become widely adopted in open-ended scenarios involving interaction with humans (e.g., chatbot), many have considered human evaluation using domain experts or crowd raters such as Amazon Mechanical Turk [129, 272] to examine models’ response quality. As an alternative to human labeling, previous work has shown that LLM-as-a-judge can be effective human preference proxies (e.g., AlpacaFarm [66], MT-bench [293], AlpacaEval [151]).

Benchmark leakage. A fundamental limitation of static benchmarks is the potential risk of test set leakage (i.e., contamination). Existing works [27, 226, 248, 279] have suggested

¹Our code is available at: <https://github.com/lm-sys/arena-hard-auto>

a growing risk of contamination, which undermines the reliability of benchmarks over time, motivating the need for benchmarks that are more frequently updated.

Live benchmarks. DynaBench [130] identifies these challenges and recommends creating living and continuously evolving benchmarks. Recent works LiveCodeBench [119], R2E [120], as well as the community based live evaluation, Chatbot Arena [40]. However, none of these focus on developing a pipeline for automatic benchmark curation.

4.3 How do you measure benchmarks?

Given a benchmark consists of a dataset D of prompts and scoring function f to measure the performance of model π_i on prompt p , we outline two key properties that an LLM chatbot benchmark approximating human preference should possess to provide a meaningful measurement of capabilities between models:

1. **Separability:** the benchmark should separate models with high confidence.
2. **Alignment with Human Preference:** f should agree with human preference.

While previous works have focused on alignment, separability is also a crucial consideration when comparing models of similar quality (e.g., different checkpoints from the same training run). However, achieving high-confidence separability is challenging due to limitations in prompt design and inherent variances in LLM evaluations. Overly simplistic prompts fail to distinguish between models, while the randomness in human and LLM judgments leads to inconsistent predictions. As a result, it is often difficult to confidently determine if a model’s apparent performance reflects a genuine difference in capability or merely noisy observations, highlighting a need for methods to verify whether a benchmark can reliably separate similar models.

Statistical measures like Pearson [209] and Spearman Correlations, commonly used in benchmarks such as AlpacaEval [151] to measure correlation to human preference ranking, may fail to adequately address model separability and ranking instability. In addition, these measures only provide a coarse signal of ranking correlation without quantifying the magnitude of performance differences between model pairs.

To address these issues, we develop three novel metrics: *Separability with Confidence*, *Agreement with Confidence*, and *Pair Rank Brier Score*.

Separability with Confidence quantifies the benchmark’s confidence by measuring its consistency in predicting the winner of a model pair across random seeds through bootstrapping. This is done by calculating the percentage of model pairs that have non-overlapping confidence intervals of their benchmark scores. A higher percentage indicates that the benchmark is more confident in distinguishing between the performance of different models, as the confidence intervals of their scores do not overlap.

Agreement with Confidence Interval measures how well benchmarks A and B confidently distinguish between two models with the same ordering. Given two models π_1, π_2 to be evaluated on A and B, we assign the following scores:

1. If both A and B can confidently separate π_1, π_2 , we assign a score of 1 if the rankings match across A and B, otherwise -1.
2. If either A or B cannot separate π_1, π_2 with confidence, we assign a score of 0.

The final agreement score across models is the average score across all unique model pairs. An agreement score of 1 implies that benchmark A confidently agrees with benchmark B on the preference of every unique model pair. Conversely, a score of -1 indicates that A confidently disagrees with B on every unique model pair.

Pair Rank Brier Score further assesses an LLM benchmark’s capability to predict the ranking of a pair of competing models by rewarding confidence in correct predictions while penalizing confidence when incorrect. Consider two models $\pi_1 > \pi_2$ with disparate quality. Although two benchmarks A and B predict the same ranking $\pi_1 > \pi_2$, they predict $P(\pi_1 > \pi_2)$ as .60 and .90, respectively (undetectable by Spearman correlation). These benchmarks would result in very different Brier scores, reflecting their ability to quantify the magnitude of performance difference between the models. If both benchmarks give the wrong prediction of the winner, we prefer the benchmark with a less confident prediction. Brier score weighs a benchmark’s accuracy and its ability to quantify the appropriate level of uncertainty in its predictions.

For each model pair, the benchmark’s forecast probability $f(\pi_i < \pi_j)$ can be estimated as $P(f(\pi_i) < f(\pi_j))$ with $f(\pi_i) \sim \mathcal{N}(\mu_i, \sigma_i^2)$ and $f(\pi_j) \sim \mathcal{N}(\mu_j, \sigma_j^2)$, where μ_i and σ_i^2 are the bootstrapped mean and variance by evaluating multiple times. The Brier Score Loss is then calculated over the benchmark’s prediction for each model pair with respect to the ground truth probability, where Brier Score is defined as $\frac{1}{N} \sum_{\{i,j\}} (f(\pi_j < \pi_i) - O(\pi_j < \pi_i))^2$, where N is the number of model pairs and O is the ground truth outcome.

Let scoring function $s(\pi_i)$ output the bootstrapped benchmark score for model π_i . Since $s(\pi_i)$ is the bootstrapped score, we can consider it as a random variable $s(\pi_i) \sim \mathcal{N}(\mu_i, \sigma_i^2)$ where μ_i and σ_i^2 are the bootstrapped mean and variance, respectively. Now, consider model pair (π_i, π_j) : let $\hat{P}(s(\pi_i) < s(\pi_j))$ be the forecasted probability that model π_i performs lower than π_j on the ground truth measurement. We observe that since $s(\pi_i)$ and $s(\pi_j)$ are normally distributed, we can directly evaluate $\hat{P}(s(\pi_i) < s(\pi_j))$. Let $O_{\pi_i \prec \pi_j} = \mathbb{1}(\pi_i \text{ performs lower than } \pi_j \text{ on ground truth measurement})$ represent the ground truth outcome. The Brier Score Loss is then calculated over the benchmark’s prediction for each model pair with respect to the ground truth outcome O , where Brier Score is defined as $\frac{1}{N} \sum_{\{i,j\}} (\hat{P}(s(\pi_i) < s(\pi_j)) - O_{\pi_i \prec \pi_j})^2$, where N is the number of model pairs.

Separability with Confidence, Agreement with Confidence Interval, and Pair Rank Brier Score each provide useful perspectives for evaluating the effectiveness of LLM benchmarks. However, no single metric is intended to be individually sufficient. Ultimately, these metrics are complementary and should be considered together holistically. We claim that together, these metrics offer a robust framework for assessing benchmark performance, balancing the need for clear differentiation with alignment to human preferences.

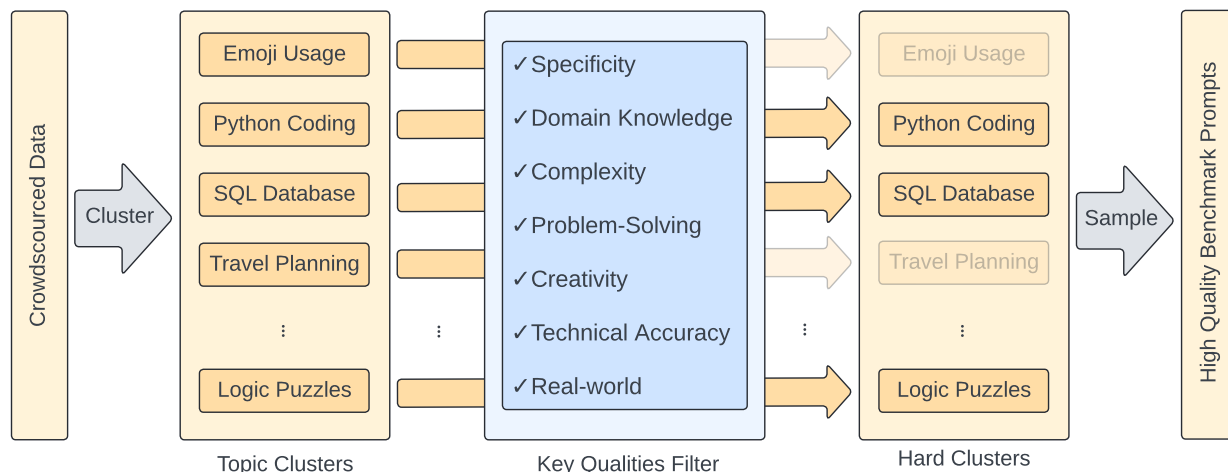


Figure 4.1: **BenchBuilder Pipeline.** Starting with a live data source of crowdsourced user prompts, we first cluster their embeddings to form topic clusters. An LLM annotator then assigns quality scores based on the required skills. Clusters with low quality scores are filtered out, and we sample from the remaining high-quality clusters to create a diverse and challenging dataset of benchmark prompts.

4.4 The BenchBuilder Pipeline and Arena-Hard-Auto v0.1 Dataset

BenchBuilder: A pipeline for converting messy data into high-quality benchmarks

BenchBuilder automatically extracts diverse, high quality prompts from a dataset of 200,000 user queries collected via Chatbot Arena. This is done by assigning a quality score to each prompt and by sampling prompts with a high quality score evenly across a wide range of topics. An outline of our data creation pipeline is shown in Figure 4.1.

Key Prompt Qualities

- **Specificity:** Does the prompt ask for a specific, well-defined output without leaving any ambiguity?
- **Domain Knowledge:** Does the prompt test the AI’s knowledge and understanding in a specific domain or set of domains?
- **Complexity:** Does the prompt have multiple components, variables, or levels of depth and nuance?
- **Problem-Solving:** Does the prompt require active problem-solving: analyzing and clearly defining the problem and systematically devising and implementing a solution?

- **Creativity:** Does the prompt require a creative approach or solution?
- **Technical Accuracy:** Does the prompt require an answer with a high degree of technical accuracy, correctness and precision?
- **Real-world Application:** Does the prompt relate to real-world applications?

To discover high quality prompts, we define seven qualities of a prompt which signify the skills required to successfully answer, including specificity, domain knowledge, complexity, and more shown in the figure above. These qualities are used by an LLM annotator to assign a “quality score” to a given prompt p , which indicates how many of the above qualities are seen in p . Full instruction prompt is in Section A.14.

To ensure that the filtered prompts represent a wide range of tasks, we adopt a topic modeling pipeline in BERTopic by first converting each prompt with OpenAI’s embedding model (text-embedding-3-small [196]), reducing dimension with UMAP, and using a hierarchical-based clustering algorithm (HDBSCAN) to identify clusters which are then summarized using GPT-4-Turbo [194].

As some topic clusters contain mostly trivial or ill defined prompt like “hi”, we select the topic clusters with mean quality score ≥ 6 out of 7 and sample prompts equally across each of these clusters. The resulting dataset consists of mostly well-defined, technical problem-solving queries as required in the above key criteria. Dataset statistics and further description of the filtering process used for Arena-Hard-Auto v0.1 along are in the following section.

Arena-Hard-Auto v0.1: A set of challenging, high-quality prompts from Chatbot Arena

We utilize the BenchBuilder pipeline to create Arena-Hard-Auto v0.1, a set of 500 high-quality prompts taken from 200,000 prompts from Chatbot Arena, an open source platform for LLM evaluation. We first remove any duplicate prompts, multi-turn conversations, or non-English conversations, and obtain over 4,000 topic clusters and 75,000 queries covering a wide range of domains.

Then, we use the GPT-3.5-Turbo as judge to assign a “quality score” to each prompt and remove prompts which either have a score < 5 or which belong to a topic cluster with a mean score < 3 . We present the statistics in Table 4.1. Notably, around 18% out of 75,000 queries are higher quality according to GPT-3.5-Turbo. We then use GPT-4-Turbo as judge to assign a quality score and ensuring the quality of the remaining prompts, resulting with over 500 high quality clusters. We randomly sample 500 prompts from these clusters, creating Arena-Hard-Auto v0.1. Additionally, we confirm our final dataset does not contain any personally identifiable information or offensive content.

In Figure 4.2, we show examples of topic clusters ranging from low to high mean scores provided by GPT-3.5-Turbo. We can observe clusters with higher scores often correlate to challenging topics or tasks for LLMs like game development or mathematical proofs. On the other hand, clusters with lower scores point to trivial or ambiguous questions like “Flirty

Texting Strategies”. We present several example prompts and their respective topic cluster below.

Cluster 1: Greetings and Well-Being Inquiry (Mean Score: 2.7)

Yo, what up my brother (Qualities: None)

Cluster 2: US Presidents Query (Mean Score: 3.2)

Who was the president of the US in 1975 (Qualities: Specificity, Domain-Knowledge, Technical Accuracy, Real-World)

Cluster 3: Physics Problem Solving (Mean Score: 5.0)

A 50,000 kg airplane initially flying at a speed of 60.0 m/s accelerates at 5.0 m/s² for 600 meters. What is its velocity after this acceleration? What is the net force that caused this acceleration? (Qualities: Specificity, Domain-Knowledge, Complexity, Problem-Solving, Technical Accuracy, Real-World)

Cluster 4: OpenCV Image Processing Technique (Mean Score: 5.5)

you are given a task to detect number of faces in each frame of any video using pytorch and display the number in the final edited video. (Qualities: All)

To see whether “quality score” assigned during BenchBuilder’s pipeline correlates with separability and agreement, we sample 50 prompts per score and compare the responses from GPT-4 and Llama-2-70b-Chat [258], with GPT-4-Turbo as judge. In Figure 4.4 (Left), we observe a strong correlation between high potential score and the win-rate of GPT-4-Turbo over Llama-2-70b-Chat. A similar trend is also observed in other model pairs such as Claude Sonnet vs Haiku and Mistral-Large [249] vs Mixtral [122].

Since qualities assigned by GPT-4-Turbo are critical to ensure the final prompt selection is high-quality, we verified its accuracy as annotator. We construct “ground truth” labels for 200 randomly sampled queries by collecting majority votes from GPT-4o [197], Claude-3-Opus [8], and Gemini-1.5-Pro [248]. GPT-4-Turbo has 85.6% accuracy against “ground truth” labels, validating its ability as annotator.

4.5 Evaluation with LLM-as-a-Judge

Evaluating models on challenging queries such as Arena-Hard-Auto v0.1 is a non-trivial task. Most queries involve deep domain knowledge and problem solving skills, requiring expert-level judgment to evaluate the answer quality. Unfortunately, it is prohibitively expensive and time consuming to use human judges. Following LLM-as-a-Judge [293] and AlpacaFarm [66], we employ LLM-as-a-judge framework to approximate human preference.

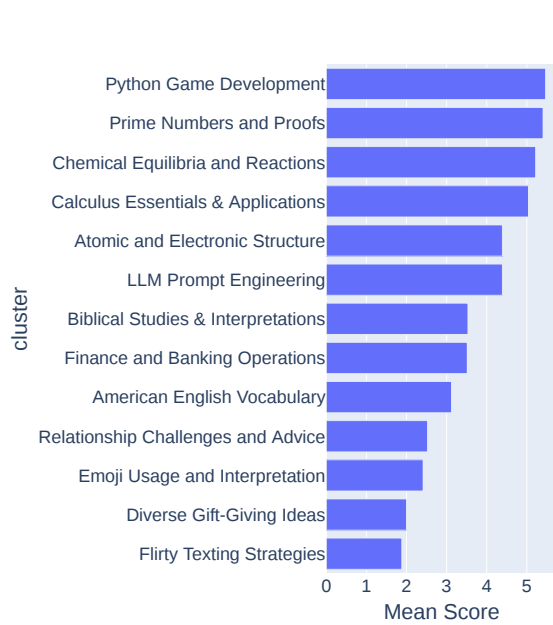


Figure 4.2: Mean score of various topic clusters in descending order. Higher-scoring clusters correlate to challenging topics, while lower scores indicate trivial or ambiguous questions. A more complete topic cluster plot is in Figure A.19f.

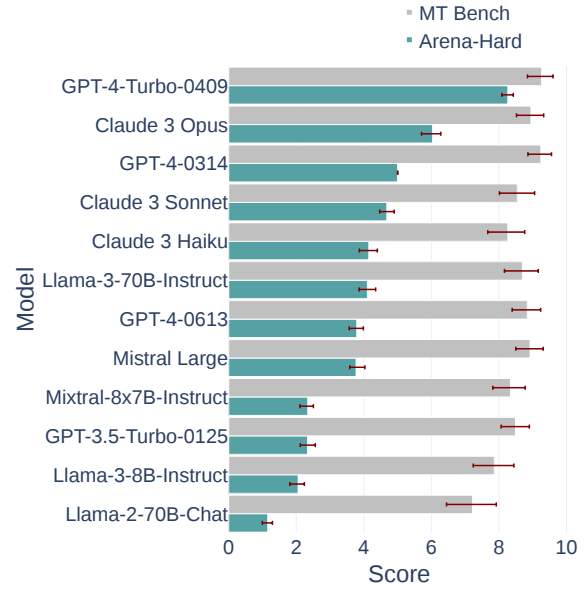


Figure 4.3: Comparison between MT-bench and Arena-Hard-Auto v0.1. The latter offers significantly better separability between models and tighter confidence intervals. GPT-4-0314 has no variance in Arena-Hard-Auto v0.1 because it is the anchor model.

To obtain a score for model π_i on prompt p , we consider the pairwise comparison setup against a strong baseline model π_{base} (GPT-4-0314) and employ a strong judge model (e.g., GPT-4-Turbo or Gemini-1.5-Pro) to measure preference. We obtain a score for each output $\pi_i(p)$ by asking the judge to rank its preference of $\pi_i(p)$ or $\pi_{base}(p)$ on a Likert scale [155] (1 = prefers $\pi_i(p)$ much less than $\pi_{base}(p)$, 5 = prefers $\pi_i(p)$ much more than $\pi_{base}(p)$). This way, a model will be penalized more in big losses than small losses, which we find to be effective in separating models. We also employ chain-of-thought (CoT) [276] to prompt the LLM judge to generate its own answer first before giving judgments. The full judge prompt can be found in Section A.14.

To avoid potential position bias, we adopt a two-game setup – per query we swap the models on the first and second position. This results in 1000 judgments per model evaluation. Following Chatbot Arena, we adopt the Bradley-Terry [24] model to produce model’s the final model scores. We aggregate all pairwise comparisons to the baseline model for all models and bootstrapping the comparisons to retrieve a bootstrapped confidence interval of all models’ win-rate against the baseline, producing a ordered ranking of all models by their win-rates.

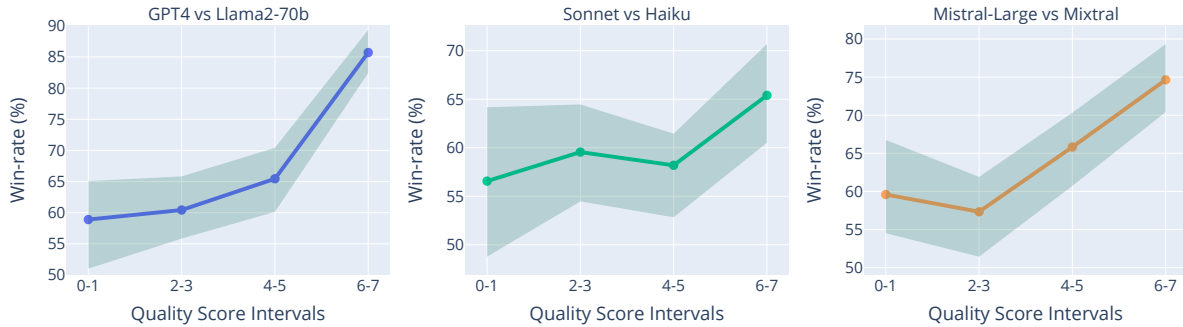


Figure 4.4: Win-rate of three model pairs (GPT-4-0613 vs Llama-2-70b-chat, Claude-3-Sonnet-20240229 vs Claude-3-Haiku-20240307, and Mistral-Large vs Mixtral-8x7b-Instruct-v0.1) over “quality score”. We randomly sample 50 queries for each quality score 0-7 and bootstrap a win-rate and confidence interval between model pairs on each score interval of 2. We observe a similar trend of win-rate between model pairs becomes increasingly separable as the quality score increases.

Quality Score	1+	2+	3+	4+	5+	6+	7+
% of queries	95.4	83.5	61.9	48.7	33.8	17.9	0.2
Qualities	Specificity	Domain-knowledge	Complexity	Problem-solving	Creativity	Tech. Accuracy	Real-world
% of queries	57.3	63.4	35.0	34.9	26.1	39.0	87.9

Table 4.1: First row is the percentage of queries with quality scores of the column or more in 75K Chatbot Arena data assigned by GPT-3.5-Turbo. Second row is the percentage of queries in 75K Chatbot Arena labeled by GPT-3.5-Turbo with each of the 7 qualities.

4.6 Experimental Results

Setup and Baselines

To compare Arena-Hard-Auto v0.1’s *separability* and *alignment with humans* against other widely used benchmarks, MT-Bench [293] and AlpacaEval 2.0 Length Controlled [67], we obtain 95% confidence interval of model performances via applying 100 rounds of bootstrapping on judgment results for each benchmark. For AlpacaEval, we use pre-existing results from their repository. We obtain MT-Bench judgment with no modification to their recommended evaluation setup. For Arena-Hard-Auto v0.1, we employ the system proposed in section 4.5 by choosing gpt-4-0314 as baseline model for pairwise comparison.

¹gpt-4-turbo-2024-04-09, claude-3-opus-20240229, claude-3-sonnet-20240229, gpt-4-0314 [193], gpt-4-0613, mistral-large-2402, qwen1.5-72b-chat [250], mistral-medium, claude-2.0, gpt-3.5-turbo-0613, claude-2.1, gemini-pro [83], mixtral-8x7b-instruct-v0.1, gpt-3.5-turbo-0314, yi-34b-chat [2], tulu-2-dpo-70b [117], dbrx-instruct-preview [252], vicuna-33b [38], starling-lm-7b-alpha [302], llama-2-70b-chat

	Chatbot Arena	MT Bench	AlpacaEval 2.0 LC	Arena-Hard-Auto v0.1
Confidence Agreement	N/A	26.1%	81.2%	89.1%
Separability	85.8%	22.6%	83.2%	87.4%
Spearman Correlation	N/A	91.3%	90.8%	94.1%
Brier Score	N/A	0.09	0.11	0.069
Real-world	Yes	Mixed	Mixed	Yes
Freshness	Live	Static	Static	Frequent Updates
Eval cost per model	very high	\$10	\$10	\$25
Prompts per model	10000+	160	800	1000

Table 4.2: We use a set of top-20 models¹ on Chatbot Arena (2024/04/13) that are also present on the AlpacaEval leaderboard to calculate separability and agreement per benchmark. We consider the human preference ranking by Chatbot Arena (English only) as the reference to calculate agreement.

Model	GPT4-T	Claude3-Opus	Gemini1.5-Pro	Llama3-70B	Ensemble-as-Judges
Confidence Agreement	89.1%	66.7%	84.8%	65.6%	91.5%
Separability	87.4%	83.68%	82.11%	81.6%	89.5%
Spearman Correlation	94.1%	77.0%	95.2%	70.5%	96.5%
Brier Score	0.069	0.170	0.064	0.196	0.065

Table 4.3: Statistics of Arena-Hard-Auto v0.1 with four LLM different judges: GPT4-T (gpt-4-1106-preview), Claude-3-Opus, Gemini1.5-Pro (gemini-1.5-pro-0514), Llama3-70B (llama-3-70b-instruct). We compare rankings produced by these judges against Chatbot Arena (English) ranking (as of 2024/04/13). We observe GPT-4T and Gemini1.5-Pro have higher agreement than Claude-3-Opus and Llama-3-70B. Furthermore, the ensemble of GPT4-T and Gemini1.5-Pro shows even higher agreement.

In order to compare these benchmarks fairly, we use a set of top-20 models¹ on Chatbot Arena [40] (2024/04/13) that are also presented on AlpacaEval leaderboard (2024/04/13) as ground truth for human preferences on the model ranking orders. In [section 4.6](#), we individually compare Arena-Hard-Auto v0.1 to Chatbot Arena within a more similar distribution of user queries. We compute separability using 95% confidence intervals and consider the human preference ranking by Chatbot Arena as the reference to calculate agreement, Spearman correlation, and Pair Rank Brier Score across AlpacaEval, MT-Bench, and Arena-Hard-Auto v0.1.

Separability and Agreement Across Benchmarks

In [Table 4.2](#), Arena-Hard-Auto v0.1 shows the highest separability (87.4%) against widely adopted LLM benchmarks and offers highest agreement (89.1%) to Chatbot Arena at \$25 cost. In [Figure 4.3](#), we show Arena-Hard-Auto v0.1 offers significantly stronger separability

Model	Arena-Hard-Auto v0.1	Random Sample 1	Random Sample 2
Confidence Agreement	84.2%	57.5%	66.1%
Separability	80.5%	74.7%	76.3%
Spearman Correlation	94.7%	64.7%	72.5%
Brier Score	0.069	0.215	0.162

Table 4.4: We compare Arena-Hard-Auto v0.1 with two sets of 500 prompts randomly sampled from 75K Chatbot Arena user queries. We evaluate the set of top-20 models and compare various statistics across. Each prompt is judged only once by positioning the baseline answer first.

	Arena-Hard-Auto v0.1
Confidence Agreement	92.0%
Spearman Correlation	96.4%
Brier Score	0.055

Table 4.5: We compare Arena-Hard-Auto v0.1 (gpt-4-1106-preview as judge) to Chatbot Arena Category Hard Prompt (English) on the same set of top-20 models*. By comparing Arena-Hard-Auto v0.1 to a similar distribution of queries from Chatbot Arena, we obtain even higher alignment to human preferences.

against MT Bench with tighter confidence intervals. With only 500 prompts, Arena-Hard-Auto v0.1 achieve impressive alignment to and even higher separability than Chatbot Arena Ranking, which constitutes for approximately 1 million real-world human preferences.

Notably, we observe a significant gap between MT-bench’s Spearman Correlation (91.3%) and confidence agreement (22.6%) to Chatbot Arena, an example where Spearman Correlation fails to account for variance of the rankings, and hence cannot adequately measure important ranking granularity of top LLMs. We present a visual comparison between Arena-Hard-Auto v0.1 and MT-Bench in [Figure 4.3](#), highlighting Arena-Hard-Auto v0.1’s improved separability.

Comparing to a similar distribution of human preference

We set Chatbot Arena Category Hard Prompts (English) as ground truth reference. Chatbot Arena Category Hard Prompt is a human preference ranking leaderboard based on votes from a harder subset of 200,000+ prompts extracted from over 1 million Chatbot Arena queries using a similar method as BenchBuilder. In [Table 4.5](#), we show Arena-Hard-Auto v0.1 achieving an overall higher alignment (92.0% Confidence Agreement) to human preferences.

Comparing to Random Subsets of Arena

To further verify BenchBuilder pipeline’s ability to select high-quality prompts, we compare Arena-Hard-Auto v0.1 to two sets of 500 randomly selected prompts from the 75,000 user queries prior to pipeline extraction. We compare their ranking preferences of the set of top-20 models produced by GPT-4-1106-Preview as judge. In Table 4.4, we observe Arena-Hard-Auto v0.1 is significantly better than both random baselines, verifying BenchBuilder’s effectiveness at selecting superior prompts for a LLM benchmark.

Ensemble-as-Judges

We study four popular LLMs² as judge on Arena-Hard-Auto v0.1. We statistically compare their ranking preference of the set of top-20 models¹ to the Chatbot Arena Leaderboard in Table 4.3. Out of the four judges, GPT-4-1106-Preview has the highest separability and agreement to human ranking preference. We also observe GPT-4-1106-Preview and Gemini-1.5-Pro have much higher agreement than Claude-3-Opus and Llama-3-70b-instruct, with more than 20% improvement.

We also experiment combining GPT-4-1106-Preview and Gemini-1.5-Pro as an ensemble judge by combining their judgment results. This achieves an even higher separability and agreement to human ranking preference, with agreement reaching 91.5%, demonstrating effectiveness of judging LLMs with Ensemble-as-Judges.

4.7 Limitations

Although our data source comes from diverse distributions, we acknowledge possible biases within our pipeline. For example, the 7 qualities we defined might not be all-encompassing and could potentially bias towards a limited distribution of prompts, such as technical fields. Further, prompt selection process could be biased by LLM annotators. Additionally, for model evaluation, we are employing LLMs as judges, who are known to suffer from verbosity bias [67, 293]. In Table A.12, we present how verbosity can effect LLM as Judge on Arena-Hard-Auto v0.1. We also acknowledge potential self-bias in LLM judges (e.g., GPT-4 Judge prefers GPT-4 answers) in Table 4.3. Arena-Hard-Auto v0.1 also does not evaluate multi-turn or non-English interactions.

In the future, we plan to further the exploration of a more generalized and diverse set of prompt qualities, establishing a pipeline to systematically produce benchmark across a more comprehensive and inclusive distribution of prompts while maintaining high separability and alignment to humans. Additionally, we believe our approach of utilizing LLM-as-a-judge can be improved, such as developing a system of Ensemble-as-Judges, similar to Judges with Jury [262], to reduce biases and improve correlation.

²GPT-4-1106-Preview, Claude-3-Opus-20240229, Gemini-1.5-Pro-0514, and Llama-3-70b-instruct

4.8 Conclusions

We introduced BenchBuilder, a data curation pipeline that transforms crowdsourced data into high-quality benchmarks by seven key qualities. This pipeline enables building challenging and evolving benchmarks which is crucial for evaluating today’s advanced language models. Our evaluation metrics, including separability and agreement with confidence, provide a comprehensive assessment of benchmarks. We show the resulting benchmark, Arena-Hard-Auto v0.1, significantly improves separability and alignment with human preferences over existing benchmarks, achieving 89.1% agreement with Chatbot Arena rankings at only \$25 per evaluation. We expect Arena-Hard-Auto v0.1 to be useful for LLM developers to evaluate their models with confidence and BenchBuilder to be a valuable tool for developers seeking to extract high-quality benchmark from vast amounts of data with minimal human effort.

Chapter 5

VisionArena: 230K Real-World User-VLM Conversations

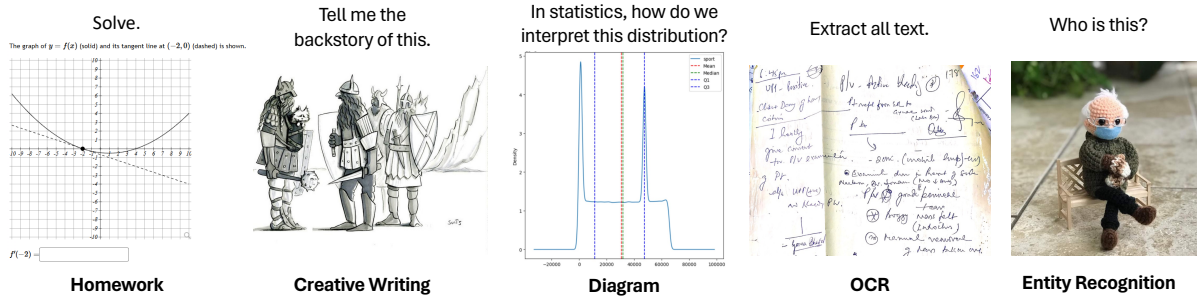


Figure 5.1: **VisionArena overview.** VisionArena collects 230K real-world conversations between users and vision-language models, with preference labels from human voters.

5.1 Introduction

Visual language models (VLMs) [4, 8, 193, 218] are being increasingly used in a wide range of real-world applications including image captioning and story telling, document understanding, web development, and embodied systems. While these models have made remarkable progress on a wide range of benchmarks [9, 89, 96, 97, 283, 287], existing VLM benchmarks focus largely on static, single-turn tasks with predetermined correct answers, overlooking the open-ended, evolving nature of real-world user interactions. They also rarely capture multi-turn dialogue, incorporate diverse context, or reflect the fluidity of user intent. As such, they provide a simplified snapshot of VLM capabilities.

Understanding these real-world interactions across a variety of tasks is essential for developing models that align with human expectations and perform effectively. To address this, previous works such as Chatbot Arena [40] and WildVision [167, 168] crowdsource evaluation by hosting platforms where users can freely interact with pairs of VLMs and provide preference votes. Building off of these works, we introduce **VisionArena**, a dataset of 230K real-world conversations between

Dataset	# Convs	# Models	# Users	# Langs	% Unique Images per Sample	Avg. # Turns	Avg. # Tokens per Prompt	Avg. # Tokens per Response	Human Preference
LMSYS-Chat-1M	1,000,000	25	13,500	35	-	2.0	36.9	214.2	No
WildVision-Battle	10,383	19	-	28	56.2	1.2	57.8	131.7	Yes
WildVision-Chat	45,170	9	-	26	33.4	1.4	81.3	171.6	No
VisionArena-Battle	30,000	17	14,031	90	76.4	1.3	90.2	393.6	Yes
VisionArena-Chat	200,000	45	72,933	138	62.1	1.5	184.1	634.3	No

Table 5.1: **Dataset Comparison.** Compared to previous VLM preference benchmarks, VisionArena contains 3x the amount of data, with more users, language, models, unique images, and conversation turns.

users and 38 VLMs in 135 languages, collected through the Chatbot Arena platform. VisionArena consists of:

- **VisionArena-Chat:** 200,000 single and multi-turn chat logs between users and VLMs, spanning 138 languages, 73k users, and 45 open source and proprietary VLMs.
- **VisionArena-Battle:** 30,000 conversations where users interact with two anonymized VLMs, along with preference votes indicating which response they prefer.
- **VisionArena-Bench:** An automatic benchmark consisting of 500 diverse user prompts that can be used to cheaply approximate model rankings via automatic benchmarking with VLM as a judge.

We conduct analysis of these datasets and construct a set of popular question categories including captioning, OCR, humor, creative writing, entity recognition, and diagram understanding. We also explore the influence of stylistic properties of responses such as response length, markdown, and specificity on human preference. We find that more open-ended questions like captioning and humor are heavily influenced by style, which causes certain models like InternVL to have a disproportionately higher ranking in these categories. We provide this metadata with VisionArena to enable further analysis. We also highlight common failure modes of VLMs and provide a small curated set of user prompts where top proprietary models fail, including complex spatial reasoning and planning tasks.

Next, we demonstrate how VisionArena can be used to improve VLMs through instruction finetuning. Compared to LLaVA-Instruct-158K [161], by finetuning on data from VisionArena-Chat, models show a 17 point improvement in MMMU [283] and a 46 point improvement on the human preference benchmark WV-Bench [168]. In addition to VisionArena, we also release this finetuned model. Lastly, we build on existing work in automatic benchmarking of VLMs to show that evaluating on the 500 prompts in VisionArena-Bench results in a model lineup that is consistent with the much larger online preference leaderboard Chatbot Arena. When compared with other automatic preference benchmarks like WildVision-Bench, VisionArena-Bench is far more predictive of the online Chatbot Arena VLM leaderboard performance, which contains over 100,000 user votes as of October 23rd, 2024. We believe that VisionArena is a valuable resource to better understand how people are currently using VLMs and will be the foundation for research in VLM development and evaluation. In the future we plan to continue regular data releases including a large variety of models and multi-image conversations.

5.2 Related Works

Crowdsourced Evaluations. In the past few years, several platforms have emerged that aim to crowdsource evaluation for LLMs and VLMs by allowing users to provide preference votes. These platforms, such as Chatbot Arena [40], allow anyone to freely engage in open-ended conversations with state-of-the-art commercial and open-source models. Users are able to directly chat with specific models or chat with pairs of anonymous models side-by-side. In the anonymous side-by-side mode, users can provide direct feedback on which responses they preferred, which is used to build a leaderboard. WildVision [168] adopts a similar style to Chatbot Arena except that users interact with VLMs instead of LLMs. Our platform builds upon these works by creating a unified interface that allows users to chat with *either* LLMs or VLMs.

Public Chat Datasets: LMSYS-Chat-1M [291], OpenAssistant [134], and WildChat-1M [56, 134] are all public datasets constructed by capturing users conversations with state-of-the-art LLMs. These datasets have been highly influential because they represent more natural human conversations and often contain reward feedback signals. However, because these datasets capture text-only conversations, they do not provide insight into how users incorporate images into conversations and the behavior of VLMs. Building on the success of public chat datasets, there is a recent effort to extend the public chat datasets to the visual domain with WildVision, a dataset of 45k chat logs with 9 visual question answering models and 10.4K battle logs across 19 models. In contrast, our VisionArena data set contains 200K chat logs and 30K battle logs across 40+ models, including all of the strong proprietary models and many open-source models, making it the largest and most complete VLM conversation dataset to date. See Table 5.1 for dataset comparison.

VLM Benchmarks: Currently, VLM benchmarks are typically static datasets that have closed-ended ground truth answers (either multiple-choice or predefined-string). Some popular examples of these benchmarks include MMMU [283], DocVQA [177], MME [78], and VQA 2.0 [90]. To combat against static nature of datasets and minimize test-set contamination, live benchmarks are also available. For example, LiveXiv incorporates updated ArXiv manuscripts for VQA [235]. To incorporate benchmarking on open-ended responses, there has been a trend towards using strong models (e.g. GPT-4o) for VLM-as-a-judge to approximate human preference. Some notable examples include WildVision-Bench [168] and Prometheus-Vision [143]. We similarly adopt the VLM-as-a-judge framework to create VisionArena-Bench, curated from questions from VisionArena, allowing it to be crowdsourced, open, and live. We show that using VisionArena-Bench, we achieve better correlation and agreement with the VLM leaderboard on Chatbot Arena, which itself has 100x more votes.

5.3 Dataset and Platform

Interface

VisionArena-Chat and VisionArena-Battle were collected from Chatbot Arena [40], an open-source platform for evaluating large language models by human preference. On our platform, users are able to directly chat with specific models (direct chat) or chat with pairs of anonymous models side-by-side (battle mode). In battle mode, users can provide direct feedback on which responses they preferred, which is used to build a leaderboard. We refer to these anonymous side-by-side

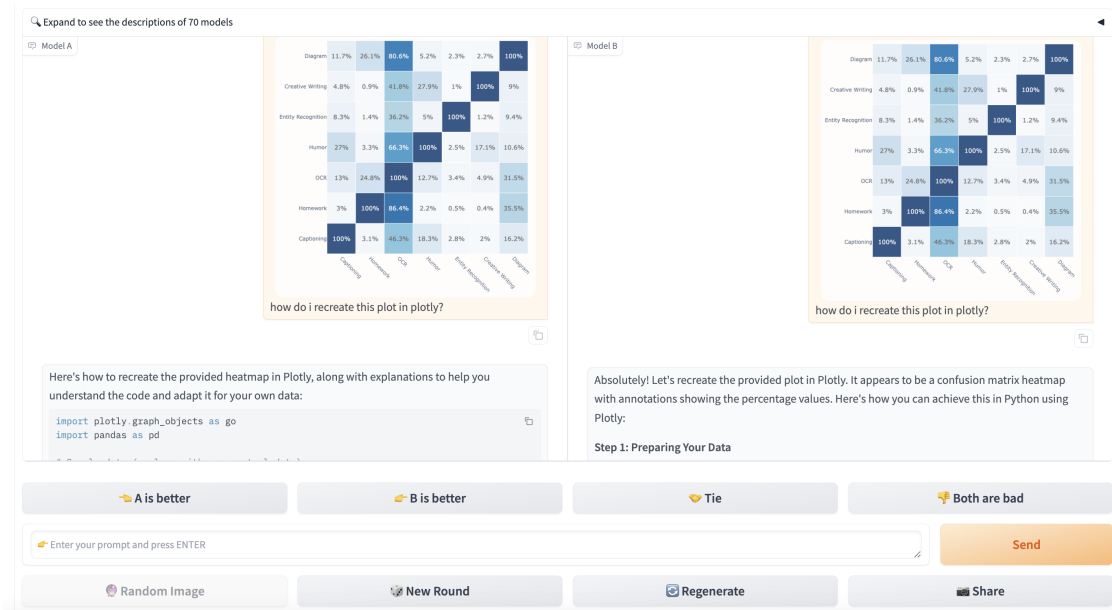


Figure 5.2: **Interface for anonymous side-by-side chat.** Users interact with two anonymous VLMs simultaneously and vote for their preferred response.

chats as ‘battles’. An example of our interface can be found in the supplemental. Unlike previous VLM crowdsourcing platforms, we integrate LLMs and VLMs into one unified chat interface with a simple routing mechanism. In side-by-side chat, if a user uploads an image in the first turn of their conversation, we automatically select from two available VLMs; otherwise we sample from the available LLMs. We then collect the votes from the image conversations to compute the VLM leaderboard. We believe this encourages users who may have initially been interested in interacting with LLMs to interact with VLMs as well. In [section 5.4](#), we show that our conversations do have important distributional differences with WildVision.

Before using our service, users must accept terms of use, giving us their consent to store and release the conversation data. The platform is free to use, and there is no registration process. We are supported by sponsorships with inference providers. VisionArena is a subset of conversations collected from February 2024 to September 2024. Given the language and question distribution of our collected conversations ([Figure A.23b](#), [Figure 5.4b](#)), the majority of our users are likely located in North America, Europe, and East Asia and work in STEM related fields.

To encourage user interaction, we provide a ‘random image’ button, which samples from a preset bank of images from 5 datasets: NewYorker [118], ChartVQA [175], DocVQA [177], TextVQA [239], and WikiArt [246]. We exclude these in VisionArena-Battle as we aim to capture the natural distribution of user inputs when computing leaderboard rankings, but we do include conversations with preset images in VisionArena-Chat, which make up around 15% of conversations.

Moderation. We apply several moderation steps before sending the prompt to the model provider and perform data cleaning procedures before releasing this data to the public. Before the user receives the response from the model, we perform (not safe for work) NSFW and (child sexual abuse material) CSAM [185] image detection and then tag and terminate conversations that contain sexual, hateful, or violent content. For battles, we also perform OpenAI text moderation [195] on

Comparing with Experts and the VisionArena

To inspect the quality of the human preference votes on our platform, we check their alignment with experts opinions. We sample 5 battles in English for each model pair and have 4 experts (PhD Students) label the battles based on their preference, creating an expert-labeled dataset of 516 responses.

We then compute BT scores using the expert labels and compare those scores with the BT scores computed directly on the VisionArena. We then compute the Pearson correlation coefficient [209] and the Spearman rank correlation coefficient [242]. We obtain a Pearson correlation of 0.88 indicating a strong linear predictive relationship between the BT scores computed by experts and those obtained from the live VisionArena. We obtain a Spearman rank correlation of 0.87 indicating high agreement in the ordering between the leaderboard rankings and the ranking obtained by our expert labelers on the small subset of data.

Additionally, for a subset of 100 battles labeled by 3 experts, we observe an agreement of 0.72 (excluding ties) and 0.56 (including ties) between users and expert annotators, compared to 0.77 (excluding ties) and 0.59 (including ties) among expert annotators themselves, further demonstrating the reliability of user votes.

What types of questions do people ask?

We perform topic modeling analysis on the VisionArena-Chat prompts. Following the BERTopic framework, We randomly sample 50K English conversations and embed the documents using CLIP-ViT-B-32, perform DBSCAN clustering, and use GPT-4o to summarize each cluster [92]. We plot our top 5 clusters in Figure 5.3b (the top 20 clusters can be found in the supplement). We find that many people use VLMs to solve math and code problems, identify paintings and geographical locations, perform data analysis on tables and diagrams, explain humorous images, and create stories based on images. Notably, VisionArena-Chat contains important use cases not seen in WildVision-Chat including coding and web UI design problems, handwritten text extraction, and diagram analysis. Manually inspecting the clusters, we also see that WildVision-Chat’s clusters are often very specific to a certain task (e.g. "Detailed Cartoon Dog Description", "Rice Leaf Disease Identification"), while VisionArena-Chat’s cluster descriptions are broader which indicates the diversity of our data. Surprisingly, the majority of our questions require OCR, and we receive a large number of homework problems and diagram understanding questions. In the following section, we construct categories for each of these major use cases.

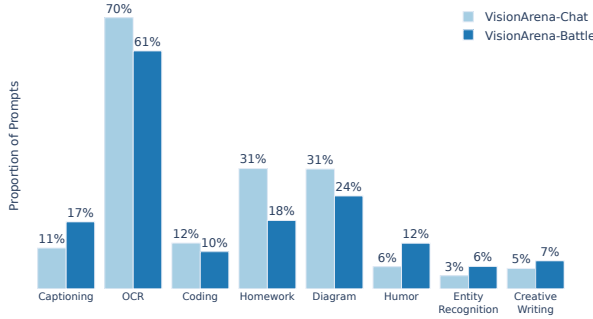
Prompt Categories

Based on the clustering analysis and manual inspection, we manually define 8 non-disjoint categories that reflect the vast majority of prompts and test different capabilities of the VLM. We use Gemini 1.5 Flash to classify each prompt, using both the image and text, into a set of predefined categories listed in Figure 5.4a. Section A.19 contains detailed descriptions of each category, the prompts used to implement the categorization with 1.5 Flash, and the correlation between 1.5 Flash category labels and those of SOTA models.

Figure 5.4b shows the distribution of category counts for both battles and direct chat conversations. We observe that direct chat conversations contain more homework and diagram under-

Category	Description
<i>Multi-Turn</i>	Multiple conversation turns.
<i>Excl. Ties</i>	Battles not ending in a tie.
<i>Excl. Refusal</i>	Neither model refuses.
<i>Captioning</i>	<i>Only</i> asks for a description.
<i>OCR</i>	Requires reading text in the image.
<i>Coding</i>	Contains a code block.
<i>Entity Recog.</i>	Identify objects, places, or people.
<i>Homework</i>	Likely a homework or exam problem.
<i>Humor</i>	Explain or compose humor.
<i>Diagram</i>	Contains a diagram (flowchart, graph).
<i>Creative Writing</i>	Story or script composition.

(a) Descriptions of VisionArena categories.



(b) **Category Distribution.** Direct chat has more coding, homework, and diagram questions; battle data has more captioning, humor, and creative writing.



(c) **Impact of confounding variables on preferences ($\hat{\gamma}$).** Length is the most influential factor, especially for open-ended questions like humor, creative writing, and captioning.

standing problems while battles contain more humor, captioning, and creative writing problems. We speculate this is because users in direct chat mode are more interested in using proprietary VLMs at no cost to assist them with their daily tasks.

Model rankings and Arena Scores for select categories and languages are shown in Figure 5.5 (full table and arena scores can be found in the supplemental). We find several interesting insights such as:

- Gemini 1.5 Pro Exp [218], InternVL [35, 36] and Reka Flash [251] achieve a worse ranking for categories with require OCR like coding, homework, and diagrams.
- InternVL has a large improvement in ranking for the captioning category. In the following section, we show that this is largely due to stylistic choices such as formatting.

	Overall	Multi-Turn	Exclude Refusal	Captioning	OCR	Entity Recognition	Coding	Homework	Diagram	Humor	Creative Writing	English	Chinese	Russian	Vietnamese
Gemini 1.5 Pro Exp	1	1	1	1	2	2	2	1	2	1	1	1	3	2	2
GPT-4o	2	2	2	3	1	1	1	2	1	2	2	2	1	3	1
Claude 3.5 Sonnet	3	3	3	2	3	3	3	4	3	3	4	3	2	1	3
Gemini 1.5 Pro	4	4	4	4	4	4	5	3	5	5	5	4	7	4	4
GPT-4 Turbo	5	5	5	5	5	5	4	5	4	4	3	5	6	5	6
GPT-4o Mini	6	6	6	6	6	7	6	6	6	6	6	6	5	6	5
Gemini 1.5 Flash	7	9	7	8	7	6	7	7	7	8	7	7	9	8	8
Claude 3 Opus	8	7	8	9	8	8	8	9	8	7	8	9	8	7	9
InternVL2 26b	9	10	10	7	10	12*	9	11	10	10	10	8	4	14	13
Claude 3 Sonnet	10	8	9	10	9	11	10	8	9	12	9	10	10	9	7
Reka Flash Preview	11	14	12	11	13	10	14	14	15	14	11	15	13	10	11
Claude 3 Haiku	12	11	11	15	11	13	11	10	11	16	12	16	11	11	10
Reka Core	13	13	13	14	12	9	12	12	12	13	13	13	14	12	12
LLAVA 1.6 34b	14	12	14	13	14	16	13	13	13	9	16	11	15	16	16*
MiniCPM v2.6	15	15	15	16	15	14*	15*	15	14	15*	14*	12	12*	15*	17*
CogVLM2 Llama3 Chat 19b	16	16	16	12	16	15*	17*	16	16	11*	15*	14	16*	13*	14*
Phi 3 Vision 128k Instruct	17	17*	17	17	17	17*	16*	17*	17	17*	17*	17	17*	17*	15*

Figure 5.5: **Model rankings across question categories and languages.** Cells with * have fewer than 100 votes. Certain models achieve a much higher ranking for a particular category, such as InternVL2 on captioning and Chinese, Reka Core on entity recognition, and Llava 1.6 on Humor. Conversely, we see certain model rankings drop, such as Reka Flash on multi-turn and diagrams, and Gemini on Chinese.

	Overall	Multi-Turn	Exclude Ties	Exclude Refusal	Captioning	OCR	Coding	Entity Recognition	Homework	Humor	Diagram	Creative Writing	English	Chinese	Russian	Vietnamese
Gemini 1.5 Pro Exp	-1	0	-1	-1	-2	0	0	0	-1	0	-1	0	-1	0	0	-1
GPT-4o	1	0	1	1	2	0	0	0	1	0	0	0	1	0	0	-1
Claude 3.5 Sonnet	0	0	0	0	0	0	0	0	0	0	1	-1	0	0	0	2
Gemini 1.5 Pro	0	0	0	0	-1	0	1	0	0	1	1	2	0	1	0	0
GPT-4 Turbo	0	0	0	-1	-1	0	-1	0	0	-1	-1	-1	0	-1	-1	-3
GPT-4o Mini	0	0	0	1	2	0	0	0	-1	0	0	0	0	1	1	-1
Gemini 1.5 Flash	0	0	0	0	1	0	0	0	1	1	0	0	0	1	0	1
Claude 3 Opus	0	0	0	0	1	0	0	0	1	-1	0	0	1	-1	0	1
InternVL2 26b	-2	-1	-1	0	-3	-1	-1	-2*	-1	-3	-3	0	-1	-1	-1	-1
Claude 3 Sonnet	1	0	1	0	1	0	1	-1	-1	3	0	0	0	-1	0	2
Reka Flash Preview	-3	-1	-3	-2	-3	-2	0	-1	-1	-2	-1	-1	-1	-1	-3	-1
Claude 3 Haiku	2	1	2	0	3	1	0	0	0	1	1	1	4	1	1	0
Reka Core	1	1	0	1	3	0	0	0	1	2	0	-1	0	1	1	1
LLAVA 1.6 34b	-1	-2	-1	-1	-2	0	0	0	-1	-3	-1	0	-3	0	-1	-1*
MiniCPM v2.6	2	2	1	2	0	2	0*	4*	2	1*	3	1*	1	0*	1*	1*
CogVLM2 Llama3 Chat 19b	0	0	1	0	-1	0	0*	0*	0	1*	-1	0*	-1	0*	1*	-1*
Phi 3 Vision 128k Instruct	0	0*	0	0	0	0	0*	0*	0*	0*	2	0*	0	0*	1*	2*

Figure 5.6: **Change in model rankings when style control is applied.** Cells with * contain battles with fewer than 100 votes. InternVL, Reka Flash, and Llava 1.6 see a ranking drop across the majority of categories, while Claude 3 Haiku and MiniCPM see an increase across most categories. The model lineup changes the most for the captioning, humor, and Vietnamese categories.

- Claude Opus, Sonnet, and Haiku [8] see an increase in ratings for multi-turn conversations.
- The Gemini class of models drops in performance on non-english conversations.

Controlling stylistic biases in evaluations

The Vision Arena captures signals from users of various backgrounds and preferences to construct its leaderboard. However, recent literature has pointed out potential confounding variables in model evaluation such as the length of the response or stylistic formatting [40, 69]. Others have

also mentioned various axes in which annotators may disagree including task underspecification, response style, refusals, and annotation errors [286]. Thus, we explore the effect of these stylistic features on the VisionArena user preference.

We follow recent work that extends the BT model to include style features [150]. Given a set of style features (e.g. response length, number of markdown headers), we add a style vector to the BT model $\vec{Z}$ where $Z_i \in \mathbb{R}^S$ is a vector of S style features. the enhanced BT model has the style coefficients $\gamma \in \mathbb{R}^S$:

$$\hat{\beta}, \hat{\gamma} = \arg \min_{\beta \in \mathbb{R}^M, \gamma \in \mathbb{R}^S} \frac{1}{n} \sum_{i=1}^n \text{CE}(\sigma(X_i^\top \beta + Z_i^\top \gamma), Y_i)$$

For each style feature Z_i , we compute the normalized difference between the feature values of both model responses. The resulting $\hat{\beta}$ represents model strengths adjusted for style effects, while $\hat{\gamma}$ quantifies the influence of style on user preferences.

To control for these stylistic factors, we modify how VisionArena computes the model scores by accounting for the stylistic differences between two answers (response length, number of markdown headers, etc) as additional features to the existing BT model.

Controlling for length and markdown. Applying style control, we see the captioning category is heavily affected by style, with Figure 5.6 showing a large difference in model rankings. We suspect this is because VLMs are heavily optimized for captioning and can usually correctly identify the main subjects and context of the image. This is supported in Table 5.2, which shows captioning questions have the smallest proportion of ‘both bad’ votes. In cases where both models provide a reasonable description of the image, the user may rely on stylistic features to determine preference.

Furthermore, models like InternVL and Reka Flash Preview see a large decrease in rankings when style control is applied. Figure A.22b shows these models have an unusually high output token count compared to models of a similar class, indicating that some models may be ‘preference hacking’ by training their models to produce long or nicely-formatted outputs. While this is not necessarily bad, it is important to consider when decoupling preference from capability.

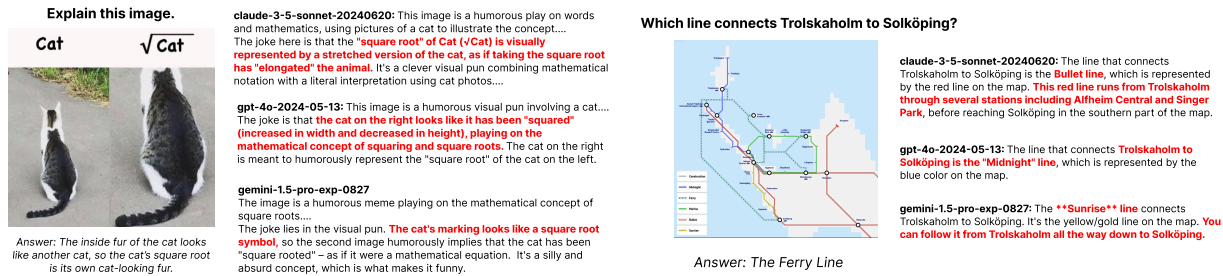
Caption	Homework	OCR	Coding	Humor	Entity Recog.	Creative Writing	Diagram
11.33	31.12	20.62	29.4	18.47	22.25	14.21	26.06

Table 5.2: **Percentage of ‘tie (bothbad)’ per category.** Captioning, creative writing, and humor categories have low percentages of bothbad responses compared to coding and homework problems.

Controlling for specificity. We further extend the Bradley-Terry model to include the effect of response specificity. We define the complexity of a response as the number of named entities in the response. We use a NER model [191] to tag each response and use the number of named entities as our specificity score. In Figure 5.4c, we see that users prefer high specificity for tasks like entity recognition and diagram understanding while placing less emphasis on specificity for tasks like captioning and homework.

Failure Cases

We use VisionArena to analyze examples which are particularly challenging for current VLMs. We first filter VisionArena-Chat for prompts where the user voted that both models are bad, and



(a) Failure case 1.

(b) Failure case 2.

Figure 5.7: VLM failure modes. The top proprietary models fail on questions which require advanced visual reasoning. For example, failure case 1 requires the visual understanding that the cat's fur patterns looks like another smaller black cat, and the linguistic connection between this and a square root. Non-truncated outputs in Section A.23.

collect a pool of 10 images which most or all of the current VLMs fail. The full set can be found in the supplemental section. Figure 5.7 shows two examples of user questions that require advanced visual reasoning. This example requires the model to (1) understand that the two cats in the image are the same (2) the pattern on the cats back gives the illusion of another cat (3) this illusion of a smaller cat in a cat can be related to the fact that the square root of a number is a smaller demonimation of that number. Figure 5.7b is an example of fine-grained spatial understanding, as the model must locate both locations and reason over the many intersecting lines in the image. Analysis of other failure cases in the supplemental section indicates that current VLMs still struggle on visual grounding tasks like reading distorted images, spatial understand and counting, as well as more complex reasoning tasks.

5.5 Instruction tuning vision-language models

Effective instruction finetuning for vision-language models depends on the diversity of instructions, the difficulty of prompts, and the quality of responses. This section demonstrates the potential of VisionArena for training high-performance instruction-following models.

We curate a high quality instruction-tuning dataset by sampling from the conversations with the highest-performing VLMs. We choose 100,000 conversations from VisionArena-Chat from the top models. This led to a dataset with conversations from 16 different models¹ including proprietary models such as GPT-4o [193], Gemini-1.5-Pro [218], and Claude-3.5-Sonnet [8] as well as open-source models such as Qwen2-VL-72B [12, 271] and Llama-3.2-90B-Vision-Instruct [4]. We compare the effectiveness of this dataset for finetuning to a 100K subset of the Llava-Instruct-158K [161].

We use Llama-3.2-11B-Vision and freeze the vision encoder while finetuning the multimodal projector and language model. We finetune for 3 epochs on the data for both our 100k dataset and the 100k Llava-Instruct dataset. In Table 5.3, we label the model finetuned on our VisionArena data Llama-3.2-VisionArena and the model trained using Llava-Instruct-158K Llama-3.2-Llava-Instruct.

¹gpt-4o-mini-2024-07-18, gpt-4-turbo-2024-04-09, gemini-1.5-pro-api-0514, claude-3-5-sonnet-20240620, gemini-1.5-pro-exp-0827, gpt-4o-2024-05-13, gemini-1.5-pro-exp-0801, gemini-1.5-flash-api-0514, claude-3-opus-20240229, gemini-1.5-flash-exp-0827, gemini-1.5-flash-8b-exp-0827, llama-3.2-vision-90b-instruct, qwen2-vl-72b [271], gpt-4o-2024-08-06, chatgpt-4o-latest-20240903, chatgpt-4o-latest-20240808

The evaluation results are shown in Table 5.3. Llama-3.2-VisionArena significantly outperforms Llama-3.2-Llava-Instruct on MME, HallusionBench, MMMU, MMMU-Pro, and WildVision-Bench. Additionally, Llama-3.2-VisionArena outperforms Llama-3.2-11B-Vision-Instruct on both MME and WV-Bench, despite being fine-tuned on 30X less data. In section A.20, we show that these improvements are not due to contamination of these benchmarks

Model	# Samples	MME[78]		HallusionBench[94]			MMMU[283]	MMMU-Pro[284]	WV-Bench[168]
		Cog.	Perc.	Acc.(all)	Fig.	Q.	Acc.	Acc.	Acc.
Llama3.2-11B-V-Instruct	3M+	<u>327.5</u>	<u>1421.7</u>	48.6	26.0	23.1	50.7	0.28	<u>47.2</u>
Llama-3.2-Llava-Instruct	100K	262.1	1067.6	38.7	13.9	8.6	27.9	0.12	10.4
Llama-3.2-VisionArena	100K	345.4	1437.0	<u>45.2</u>	<u>19.9</u>	<u>16.3</u>	<u>43.0</u>	<u>0.27</u>	56.9

Table 5.3: **Performance across models trained with different instruction tuning datasets.** Llama-3.2-11B-Vision-Instruct scores are author-reported. Fine-tuning on samples from VisionArena-Chat outperforms fine-tuning on Llava-Instruct across all benchmarks. Llama-3.2-VisionArena also outperforms Llama-3.2-11B-Vision-Instruct on both MME and WV-Bench, despite being fine-tuned on 30x less data.

5.6 VisionArena-Bench: An Automatic Offline Human-Preference Benchmark for VLMs

Lastly, we demonstrate VisionArena’s ability to cheaply approximate model preference rankings with VisionArena-Bench. Currently, online preference benchmarks like Chatbot Arena obtain a ranking for a new model by adding it to their platform and waiting days or weeks to collect enough votes for a stable ranking. For a single developer hoping to test a particular version of their model, obtaining these online model rankings is infeasible.

We develop a solution for those who need a quick and cheap evaluation of their models: VisionArena-Bench. VisionArena-Bench is a set of 500 diverse image and text prompts that accurately approximates the model ranking from the Chatbot Arena VLM Leaderboard.

Offline benchmark curation. To gather questions for this offline benchmark, we build on recent work in building benchmarks from crowd-sourced evaluations in LLM and adapt them to the context of VLM [150]. We first filter the data for single turn to prevent the user from correcting the model in its response on its second turn. We then filter out non-english conversations as we are personally unable to verify the quality of non-english prompts.

To sample diverse questions, we perform topic modeling using the library BERTopic with multimodal embeddings [92]. We extract image embeddings and text embeddings using a CLIP model (e.g. CLIP-ViT-B-32) [212]. We then average the image and text embeddings so that each document corresponds to a single embedding. Then, we use UMAP to reduce the dimensions of the

²gemini-1.5-pro-exp-0827, gemini-1.5-flash-exp-0827[218], gpt-4o-2024-05-13, gemini-1.5-flash-8b-exp-0827, internvl2-26b [36], claude-3.5-sonnet-20240620 [8], gpt-4-turbo-2024-04-09 [193], claude-3-sonnet-20240229, llama-3.2-11b-vision-instruct [4], gemini-1.5-pro-001, internvl2-4b, gpt-4o-mini-2024-07-18, claude-3-opus-20240229, gemini-1.5-flash-001, reka-core-20240501 [251], claude-3-haiku-20240307

embedding and use hierarchical-based clustering (HDBSCAN) to generate topic clusters [173, 180]. We then uniformly sample from each topic cluster to generate the 500 prompts.

Automatic evaluation with VLM-as-a-judge. To evaluate a model, we use the LLM-as-a-judge framework mentioned in [150] applied to VLMs. We first select a fixed anchor model (GPT-4-Turbo[193]) that will be used in the pairwise comparisons. To evaluate a given model M on a user prompt p , we generate responses for both M and the anchor model on p and then utilize GPT-4o as a judge to provide a preference score between the (M, anchor) pair on a 5-point Likert scale. 1 indicates a strong preference for model A and 5 indicates a strong preference for model B. We then obtain this score for all models across all prompts in VisionArena-Bench to obtain VLM-generated pairwise preference votes and use the same procedure as described in section 5.3 to produce final model scores. In section A.21, we provide the detailed judge prompt template. To avoid potential bias, we prompt the judge model to judge twice, swapping the response position between the two rounds.

To evaluate the effectiveness of our benchmark to existing work [168], we leverage standard metrics such as Spearman correlation and Kendall Tau correlation which measure the agreement between two benchmarks’ model rankings. We choose a shared set of 16 models and compare the offline benchmark rankings to the online Chatbot Arena. Table 5.4 shows VisionArena-Bench achieves a higher Spearman and Kendall Tau correlation than WildVision-Bench, with a 17.1% and 20.5% gain respectively. In Section A.21, we also compare the results using the same baseline model as WildVision (e.g. Claude-3-Sonnet-20240229), showing that the spearman correlation to Chatbot Arena’s VLM leaderboard (10/23/2024) remains the same. This demonstrates the potential of VisionArena-Bench as a cost-effective and scalable offline benchmark that closely mirrors human preferences captured in online evaluations, enabling researchers to efficiently assess and compare VLMs without the need for extensive user studies.

	VisionArena-Bench	WV-Bench
Confidence Agreement	98.6%	87.6%
Spearman Correlation	97.3%	80.2%
Kendall Tau Correlation	89.7%	69.2%

Table 5.4: **Correlation of rankings with ChatbotArena’s VLM leaderboard.** Performance comparison on 16 models² between VisionArena-Bench to Chatbot Arena’s VLM leaderboard based on confidence agreement, spearman correlation, and Kendall tau correlation. Result as of leaderboard on October 23, 2024.

5.7 Discussion, Limitations, and Future Work

Human preference benchmarks provide a critical lens for assessing performance on open-ended tasks where an explicit notion of correctness is either unavailable or subjective. Users may implicitly consider factual accuracy when making their preferences, but we would like to emphasize that **this benchmark is designed to measure human preferences rather than explicitly evaluate**

factual accuracy. We see VisionArena as complementary to existing datasets and benchmarks that measure objective correctness.

Despite the breadth of coverage offered by VisionArena, significant gaps remain in representing the full distribution of real-world use cases for vision-language models. As highlighted in [section 5.4](#), our dataset contains many sample from domains such as STEM problems, OCR tasks, and toy problems (e.g., humor and riddles). These areas, while valuable, leave critical application domains underrepresented, including geospatial applications, medical domains, and visual assistance. Furthermore, while VisionArena contains over 100 languages, many of these languages do not contain enough examples to produce a stable leaderboard. Looking forward, we hope to encourage a more diverse user base by changing our UI to be multi-lingual and improving general user experience. Lastly, we have made it easy for the community to contribute new question categories and models at <https://github.com/lm-sys/FastChat>.

Chapter 6

VibeCheck: Discover & Quantify Qualitative Differences in Large Language Models

6.1 Intro

vibe check : A process by which a group obtains a subjective assessment of another person, place, or thing. – *Urban Dictionary*

How a large language model writes a story, explains a concept, or edits an essay can be evaluated along many different dimensions such as creativity, formatting, and writing style. However, most evaluations focus on one dimension: “*correctness*”. State-of-the-art in evaluation methods remain largely focused on measuring accuracy for question answering and analytical reasoning tasks [103, 105, 266, 267], and methods which aim to provide a more holistic view of LLMs [182, 199, 289] rely on predefined concepts like conciseness, clarity, and trustworthiness to measure a model’s performance. These evaluation approaches fail to capture the open-ended nature of LLM applications and the critical dependence on subjective user preferences and context of the task. For instance, tone and creativity might be crucial in creative writing, whereas efficiency and readability are crucial in coding tasks. To best inform users of which model would be best for their needs, we require flexible evaluation methods that can both *discover* and *measure* the relevant axes to evaluate for a given task.

When interacting with a set of LLMs for an extended period, a user can often tell which model generated a particular response by looking at certain traits of the outputs. We define these identifying traits of models as “*vibes*”. For instance, users have found Llama-3 outputs tend to be more friendly compared to outputs from GPT-4 and Claude which tend to be more formal (see Figure 6.1); in other words, Llama-3 ranks high on the friendliness vibe, defined by the axis **formal** → **friendly**. Using these insights, we might select Llama for customer service tasks and Claude for coding tasks.

Understanding these vibes helps inform the development and deployment of models, but discovering and validating them for each model can be time-consuming and difficult. To address this, we outline how one can find and, more importantly, measure an LLM’s vibe by formalizing

three necessary and quantifiable traits of a useful vibe: *well-defined* (agreement among multiple users), *differentiating* (ability to distinguish between models), and *user-aligned* (predictive of user preferences).

We introduce **VibeCheck**, a system which qualitatively analyzes pairs of models by automatically finding well-defined, differentiating, and user-aligned vibes. Motivated by recent work in using LLM’s in lieu of human judgment [66, 289, 293, 296, 298], VibeCheck models the qualitative analysis process by identifying the axes on which these model outputs differ to obtain a core set of vibes (e.g. friendliness). Once these vibes are obtained, VibeCheck employs a panel of LLM judges [261] to determine where each model’s output falls on this vibe (e.g. more formal or more friendly) in order to obtain numeric scores which are then used to measure a vibe on each of our 3 key criteria.

We run VibeCheck on several datasets to evaluate its effectiveness across different scenarios in Section 6.5. First, we validate that the vibes discovered by VibeCheck align well with human-annotated differences between ChatGPT and human responses using the Human ChatGPT Comparison Corpus (HC3). Next, we demonstrate that VibeCheck outperforms a predefined list of vibes in predicting user preferences on real-world comparison data from Chatbot Arena, achieving 80% accuracy at predicting model identity and 61% accuracy and predicting user preference. Inspecting the vibes of VibeCheck, we find that Llama-70b uses more typographic emphasis, more examples, and is funnier than GPT-4 and Claude-3-Opus. Conversely, we find that GPT-4 and Claude comment much more on ethics and limitations than Llama, which is more willing to give controversial responses.

Lastly, in Section 6.6 we apply VibeCheck to several applications: text summarization on CN-N/DailyMail, math problem-solving on MATH, and image captioning on COCO. Using VibeCheck, we find insightful qualitative differences between models with similar accuracy on correctness metrics but differing user preferences. For instance, Command X prefers to add concrete intros and conclusions when summarizing in comparison to TNGL, Llama-405b often overexplains its thought process on math problems, and GPT-4 prefers to focus on the mood and emotions of the scene when captioning.

6.2 Related Work

Aspect-based evaluations. The number of benchmarks in the NLP community has exploded in recent years, with a growing body of work on exploring a more holistic evaluation of language models. Several works [13, 200, 234] aim to improve on automatic metrics like BLEU [201] and ROUGE [156] scores to better measure how well a models output aligns with the ground truth by incorporating more nuanced evaluation criteria like factual accuracy, fluency, and conciseness. Similarly, efforts have been made [130, 153, 243, 266, 267] to standardize model evaluation by evaluating models on many of these metrics across various tasks.

Moving away from measuring model outputs on ground truth responses, work from Gehrmann et al. [82], Li et al. [149], Mehri and Eskenazi [181, 182], Zhang et al. [289] evaluate model outputs on criteria like helpfulness and clarity using LLM judges on more open ended tasks like dialogue, role-play, and summarization. While these efforts supply a great foundation for measuring correctness, they all define the axes on what makes something correct beforehand. In contrast, VibeCheck aims

to automatically discover these axes (vibes) and verify their utility to the user by measuring the correlation between vibes and human preference.

Pairwise comparison of LLMs. HCI tools like Google’s AutoSxS [88] and LLMComparator [125] explores the current state of human powered LLM qualitative evaluation through interviews with data analysts. These works find that practitioners often eyeball individual examples to interpret and look at qualitative differences between the outputs of two models, and develop an interactive web based application for users to inspect side-by-side LLM outputs with an LLM based rationale as to why one output is preferred over another. While these works are focused more on software tools rather than a pipeline which can be quantitatively verified, these HCI findings inform VibeCheck’s vibe discovery mechanism to align with the human-powered qualitative process. Moreover, many NLP works [151, 166, 202, 261, 293] have explored using LLMs to predict user preference given responses from two models, showing these preference predictions often align with the judgements of human annotators. While these efforts focus more on the user experience, it does not provide an interpretable view of exactly *why* these users prefer one output over the other.

Discovering separable traits in unstructured data. In parallel to works in the machine learning community on LLM evaluation, there has been fantastic efforts in the HCI community on comparing generative model outputs as well as on using LLMs for qualitative analysis. Works like Byun et al. [26], Torii et al. [256] use LLMs to generate discussions from qualitative research data to automate the data analysis process, but note the lack of comprehensive evaluation metrics. Automated data analysis on unstructured data has also been explored in Dunlap et al. [70], Zhong et al. [296, 298], which use LLMs and VLMs to propose and validate candidate differences between two sets of text or images in the form of “set A contains more X”, and Chiquier et al. [43] employs an evolutionary algorithm to find text descriptions which best separates image classes to assist in zero-shot classification. We extend these works to pairwise inputs and introduce metrics of success which can better verify the separability, consistency, and alignment of these differences.

6.3 Vibe-based Evaluations

We define a *vibe* as an axis along which a pair of texts can differ (e.g., “formal $\rightarrow$ friendly”) that is perceptible to humans. A vibe ν is represented by a text description of the axis along with a definition of what it means to be high or low on this axis (e.g. “Tone: low = formal, high = friendly”, see Figure 6.1). Identifying vibes aids users in selecting models that best suit their specific tasks. In this work, we focus on comparing the vibes of two models by discovering the axes on which their outputs differ and quantifying the utility of these vibes.

Consider a dataset D composed of triples (p, o_A^p, o_B^p) and preference labels y_p , where p is a prompt and o_i^p are the outputs from models A and B . For each triple, a judge (human or LLM) assigns a score for vibe ν , denoted $\nu(p, o_A^p, o_B^p) \in \{-1, 0, 1\}$, which indicates whether model A scores lower (-1), similarly (0), or higher (1) than model B on this vibe. Thus, a vibe imposes an ordering on model outputs.

We define 3 key criteria of a useful vibe; it should be *well-defined*, *differentiating*, and *user-aligned*.

Well-defined: multiple evaluators agree on the ordering of outputs along the vibe. We quantify this by having two different judges (typically LLMs) compute $\nu(p, o_A^p, o_B^p)$ across dataset D and

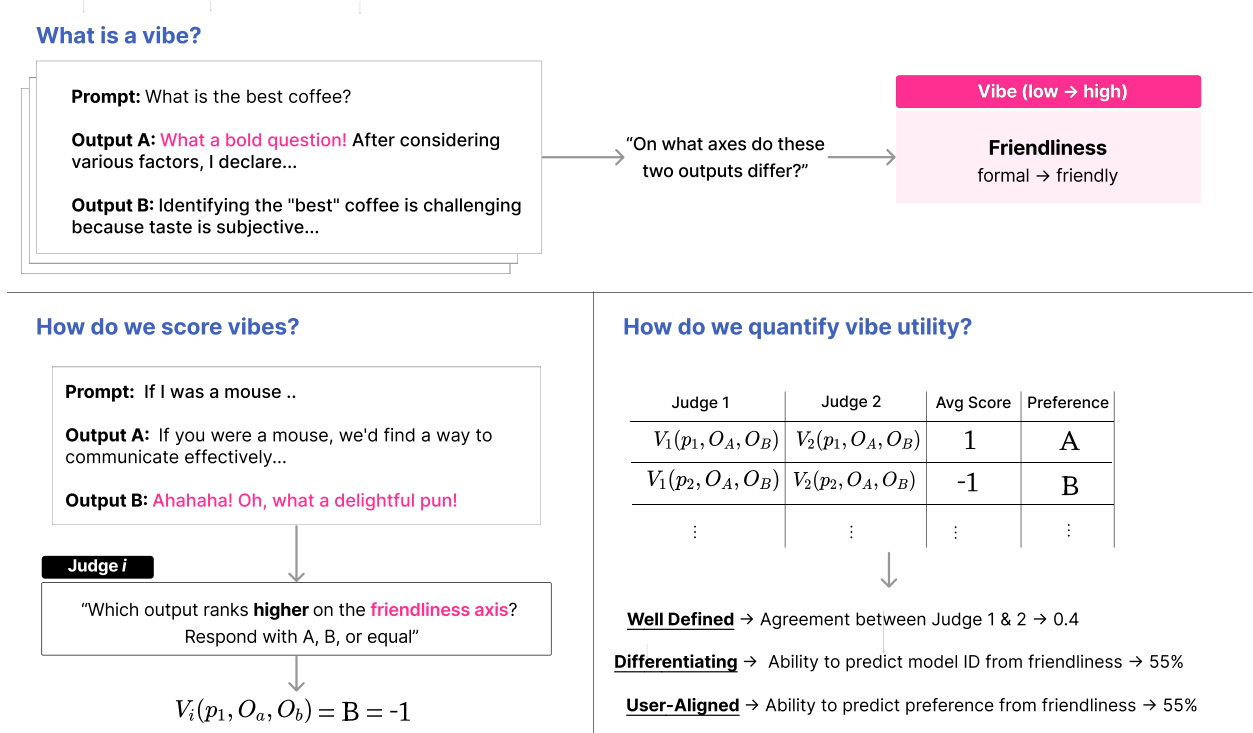


Figure 6.1: **Core components of VibeCheck.** A vibe is an axis along which a pair of outputs differ: for example, in the top panel, output A is more friendly while output B is more formal, defining a friendliness vibe. To score a prompt output triplet, a panel of LLM judges are used to determine which output falls higher on the vibe, resulting in a score of 1 (A), -1(B), or 0(tie). Finally, the scores obtained over a large set of outputs along with preference labels are used to compute vibe utility.

report Cohen’s Kappa to assess agreement.

Differentiating: one model’s outputs consistently rank higher on this vibe compared to the other’s across a set of prompts. We quantify this by calculating a *separability score* for each vibe, which measures how consistently the vibe distinguishes between the two models across all samples.

$$\text{sep_score}(\nu) = \frac{1}{|D|} \sum_{p \in D} \nu(p, o_A^p, o_B^p)$$

To measure separability across a set of vibes, we fix a pair of models (A, B) and measure the accuracy of using $\nu(o_A, o_B)$ to classify which output came from which model. We also more generally measure separability for a set of vibes $\nu_1, \dots, \nu_k$, by using $\nu_{1:k}(p, o_A, o_B)$ as a k -dimensional feature vector, then training a linear classifier to predict model A vs. model B, and reporting accuracy on a held-out set. We refer to this metric as *model-matching accuracy*.

User-aligned. One potential use of vibes is to better understand human preferences. While a vibe like “*frequent use of the letter ‘e’*” may be differentiating, it is unlikely predictive of human preferences. We assume our tuples (p, o_A^p, o_B^p) are annotated with user preferences $y \in \{-1, +1\}$, indicating which model’s output is preferred. We train a logistic regression classifier to predict y

using the same feature set $\nu_{1:k}$ as above, reporting held-out accuracy. We refer to this metric as *preference prediction accuracy*. We can measure the influence of a single vibe on preferences by examining the coefficients and p-values of the preference prediction model.

VibeCheck automatically finds high-scoring vibes across the three criteria through an iterative process: (1) discovering vibes, (2) computing their scores, (3) selecting those meeting all criteria, and (4) focusing on tuples (p, o_A^p, o_B^p) where existing vibes fail to differentiate the two models. We repeat this process to extract new, more distinguishing vibes, thus optimizing for the three key criteria while continuously refining the set of vibes.

6.4 VibeCheck

VibeCheck consists of 3 stages: vibe discovery, vibe validation, and vibe iteration. Further details on the method implementation and prompts used are located in the Section A.27.

Vibe discovery. Similar to how a data scientist would inspect a subset of examples to discover qualitative differences in outputs, we discover vibes by having an LLM (GPT-4o [197]) examine the differences seen in a random subset of d prompt triplets. We first split the d prompt triplets into smaller batches of size $batch$ and prompt GPT-4o to find differences between model A and model B across the set $\{(p_1, o_A^1, o_B^1), \dots, (p_{batch}, o_A^{batch}, o_B^{batch})\}$. To encourage the vibes to be well-defined and user-aligned, we prompt GPT-4o to generate differences that are human-interpretable and informative for understanding the overall behaviors of A and B . Below is a paraphrased system prompt used by the proposer.

You are a machine learning researcher analyzing outputs from two LLMs on the same input, identify differences along specific, mutually exclusive, and clearly defined axes that are easily interpretable by humans. For each axis, provide a concise description of what it means for an output to be "Low" and "High" on this axis.

An example axis generated in this step might be ‘Tone: Low: formal; High: friendly’. We repeat this proposal step for $\lfloor d/batch \rfloor$ sets of triplets, obtaining a final set of vibes $\{\nu_1, \dots, \nu_M\}$ by taking the union of the vibes generated in each batch. We found that GPT-4o generates 5-10 axes of variation (vibes) for each sample, so we summarize vibes across all samples in $D_{discovery}$ to find a set of K vibes which appear most often in $\{\nu_1, \dots, \nu_M\}$.

Vibe validation. Given a vibe ν from the discovery phase, we first apply each vibe to a set of validation tuples, then use this validation set to score vibes and compute inter-annotator agreement, model-matching accuracy, and preference prediction accuracy and filter out vibes with low scores.

To apply vibes on the validation set, we assign a score to each pair of outputs $\nu_j(p, o_A^p, o_B^p) \in \{-1, 0, 1\}$, indicating whether model A scores lower (-1), similarly (0), or higher (1) than model B on the vibe. A score of 0 is assigned if the outputs are equal on this vibe or if the vibe is not applicable (e.g., the vibe is about coding style but neither output contains code); otherwise, we compute the score using a set of LLM judges (GPT-4o-mini [197] and Llama-3-70b [4]). We average the score of the 2 judges and then round to -1, 0, or 1 (so 0.5 is rounded to 1 and -0.5 to -1). To avoid position bias [293], we run each LLM judge twice on each sample, swapping the order of the

outputs. If the judge’s decision is dependent on the position of the output, we deem this pair of outputs as having a similar vibe and assign a score of 0 for that judge.

Next, we use these scores to quantify each vibe on our 3 criteria and filter out any which are not well-defined, differentiating, and user-aligned. We ensure each vibe is well-defined by computing the inter-annotator agreement (Cohen’s Kappa) for each ν_j across $D_{\text{validation}}$ and remove any with Cohen’s Kappa less than 0.2, which indicates a weak agreement among judges. To ensure each vibe is differentiating, we compute the separability score and discard any vibes with a score below 0.05. As we explicitly prompt the model to produce vibes which provide useful insights into the behavior of language models, we assume these vibes are already aligned with users. Using the remaining k features, we run logistic regression using the scores $\nu_{1:k}(\mathbf{p}, \mathbf{o}_A, \mathbf{o}_B)$ as features to obtain our model matching and preference prediction models.

Vibe iteration. The filtered vibes generated in the initial vibe discovery set may not capture all the differences that contribute to user preference, resulting in a low model matching and preference prediction accuracy. We address this by iteratively refining our vibes based on tuples $(\mathbf{p}, \mathbf{o}_A^{\mathbf{p}}, \mathbf{o}_B^{\mathbf{p}})$ that were misclassified by our prior differentiation stages. Specifically, we take the prompt output triplets that were misclassified by the model matching model and ask an LLM to find new axes on which these misclassified prompts vary, which are also not represented in the current set of vibes. We then perform the same summarization/reduction procedure as before, run vibe validation/filtering, and append the resulting new vibes to the existing set of vibes. We repeat this process for a fixed number of iterations i . In practice we find that after 3-5 iterations the discovery process does not find any additional vibes that significantly reduce the error rate of the model matching predictor.

6.5 Results

We first validate VibeCheck by comparing its discovered vibes to those identified by human annotators in Section 6.5. Next, we evaluate VibeCheck on real-world user-LLM conversations with pairwise preference data, measuring how well-defined, differentiating, and user-aligned a vibe is through inter-annotator agreement, model matching accuracy, and preference prediction accuracy on a heldout set. In Section 6.5 compare the discovered vibes’ performance against an predefined list of common qualitative analysis criteria. Lastly, in Section 6.6, we demonstrate VibeCheck’s broader applicability by analyzing model differences across summarization [106], mathematical reasoning [105], and image captioning [33, 158].

Experimental setup. Unless otherwise stated, we run VibeCheck for 3 iterations, use a proposer batch size of 5, and set $D_{\text{discovery}}$ to be 20 samples per iteration. Some datasets such as MATH, CNN/DailyMail, and COCO captions have no pre-computed preference labels; to simulate preferences, we apply LLM-as-a-judge and ensemble GPT-4o and Claude 3.5 Sonnet as a judge using a similar procedure to [293], removing any samples declared a tie. Additional details on the experimental setup and hyperparameters are given in the Section A.24.

We compute average Cohen’s Kappa, model matching accuracy, and preference prediction accuracy on the top 10 vibes generated by VibeCheck on a held-out set of prompt tuples with preference labels. To obtain the top 10 vibes, we apply least-angle regression on the full set of vibes returned by VibeCheck to predict model identity, then sort by the separability score. The full list of vibes

discovered, LR coefficients and p-values from the model matching and preference prediction models, Cohen’s kappa per vibe, and separability scores are in the Section A.30.

List of predefined Vibes. As a baseline, we prompt GPT-4o to generate a set of 10 vibes shown in Figure 6.3 and Table A.23 which represent common axes on which LLM outputs differ.

Measuring VibeCheck’s alignment with human discovery

We compare the findings from VibeCheck to findings obtained via human discovery to ensure that the vibes discovered and measured by LLM’s align with humans. We utilize previous work [95], which collects responses written by humans and GPT-3.5 [233] for the same list of questions and then recruits 200 annotators to look at 100-200 prompt output triples presenting the characteristics of both human responses and ChatGPT answers. This results in a set of 10 insights (vibes) which are listed in detail in Section A.25.

In Table 6.1 we show a summarization of the top 10 vibes found by VibeCheck along with the corresponding insight found by humans which align with each vibe meaning. We see that VibeCheck uncovers most of the same vibes as the human annotators, aside from (1) GPT fabricates facts and (2) GPT focuses on a literal interpretation of the question while humans address different aspects of the question and can infer hidden meaning. The inability to find these vibes is likely a weakness of our GPT proposer, as these vibes relate to the inherent weaknesses of GPT. The complete table of VibeCheck outputs is located in Figure A.36a.

VibeCheck Vibes	Human Discovered Vibes
Humans include more references and citations	Humans include detailed citations of papers and books.
GPT is more formal/academic, humans are more casual/conversational	GPT answers are typically formal, humans’ are more colloquial.
GPT includes disclaimers about advice limitations	GPT refuses to answer questions outside its knowledge.
GPT is cautious to give advice, emphasizes seeking professional help	GPT shows less bias and harmful information.
GPT has cohesive, fluid responses with clear sentence structure	GPT writes in an organized manner with clear logic.
GPT is strictly informative, humans include personal anecdotes	GPT gives objective answers, humans use subjective expressions.
GPT has less emotional engagement, humans acknowledge emotions	GPT expresses less emotion, humans convey their feelings.
GPT has longer, more informative responses	GPT has longer, more detailed responses.
GPT has more thorough & detailed responses	GPT has longer, more detailed responses.
GPT has more comprehensive responses	GPT has longer, more detailed responses.
—	GPT is strictly focused on the question; humans diverge and shift topics.
—	GPT may fabricate facts.

Table 6.1: **Comparison of VibeCheck vibes to human labels.** Complete table in Figure A.36a. We see that the vibes discovered by VibeCheck closely align with vibes found through human analysis.

Describing user preference on Chatbot Arena

On April 18th 2024, Meta released their open-weight large language model Llama 3. On Chatbot Arena [40], a popular human preference benchmark, Llama-3-70b achieves similar preference scores to models like GPT-4 and Claude 3 Opus, despite these models outperforming Llama on traditional benchmarks like MMLU. This has led to speculation on whether there are qualitative properties of Llama that make it popular among users [68].

On Chatbot Arena, we run VibeCheck on a set of combined battles (pairwise comparisons) between Llama-3-70b VS GPT-4 and Llama-3-70b VS Claude3-Opus¹ under three settings: using the entire dataset, and using 2 subsets of the data: STEM prompts (including coding) and Writing prompts, which include creative writing, humanities questions, and general chatting. We obtain these subsets by using GPT-4o-mini to categorize the questions as a STEM Question, a Writing/Chatting prompt, or neither. The size of each subset can be found in Section A.24.

We compare the vibes found by VibeCheck to a list of predefined vibes (Table A.23) of common differences between language models which a user may be interested in. Table 6.2 shows that VibeCheck achieves higher model matching accuracy than the predefined vibes all categories and more iterations improve model matching and preference prediction accuracy. Furthermore, Figure 6.2 shows that the vibes are more fine-grained. We summarize our other findings below:

Comparing MM and PP accuracy across topics. Table 6.2 shows that MM and PP accuracy is lower for STEM questions compared to writing or overall prompts. We suspect this is because Llama’s qualitative traits (friendliness, humor, safety, etc.) are less relevant for objective questions like coding and math, and user preferences here are influenced more by factual accuracy than stylistic traits. Conversely, VibeCheck best predicts preferences for writing-oriented prompts, as style is often more important for these open ended tasks.

To understand how user preferences for these vibes vary across task domains and contexts, we analyze separability scores and preference prediction coefficients for predefined vibes in Figure 6.3. For writing tasks, formality, humor, and expressive emotional content positively correlate with user preference, while these traits negatively correlate with STEM tasks, where logical rigor is the most influential on preference. Conversely, logical rigor has minimal impact on preferences for writing tasks. While our dataset does not directly compare individual judgments, treating STEM and writing task users as distinct groups provides preliminary evidence of task-specific preferences. Additionally, lower separability scores for STEM tasks indicate less stylistic divergence in model outputs for objective questions like coding and math, making model identity harder to predict, consistent with Table 6.2.

Notable vibes for Llama-3 70B. The top 10 vibes uncovered by VibeCheck (Figure 6.2) highlight Llama’s use of formatting, willingness to engage with sensitive topics, less emphasis on ethics, and a conversational, humorous style. Finer-grained vibes include Llama’s use of bold/italics to emphasize points and increased use of personal pronouns, with ‘I,’ ‘we,’ and ‘you’ appearing 3x more in Llama outputs than GPT/Claude conversations. The preference prediction coefficients in Figure 6.2 show Chatbot Arena users tend to prefer outputs which are less focused on ethics, employ markdown and typographic emphasis to highlight key points, and employ humor to engage the user, all of which are vibes which llama possesses. We believe that this correlation between

¹Data: <https://huggingface.co/datasets/lmarena-ai/Llama-3-70b-battles>

Vibe (low -> high)	Sep Score [-0.4,0.4]	PP Coef [-0.5,0.5]	Cohen
Language and Tone. Professional, straightforward tone. -> Enthusiastic, friendly tone.			0.51
Typographic Emphasis. Minimal use of typographic emphasis, letting the text stand alone. -> Uses typographic emphasis like bold or italics to highlight key points.			0.64
Interactivity. Provides information passively without engaging the user. -> Encourages user interaction, such as posing questions or suggesting actions.			0.44
Formatting Completeness. Responses are minimally formatted, relying on plain text. -> Responses include comprehensive formatting, such as Markdown or additional stylistic elements.			0.57
Examples and Illustrations. Minimal examples. -> Provides multiple examples.			0.61
Use of Humor. Maintains a serious tone without humorous elements. -> Employs humor frequently to engage the reader.			0.62
Use of Personal Pronouns. Rarely or never uses personal pronouns. -> Frequently uses personal pronouns (I, we, you).			0.32
Ethical Consideration. Provides factual information without commenting on ethics. -> Offers ethical considerations in its responses.			0.53
Humility. Projects confidence and completeness without discussing limitations. -> Frequently acknowledges limitations in the response or areas of uncertainty.			0.41
Formality Level. Uses informal or conversational language. -> Uses formal language and expressions.			0.45

Figure 6.2: **Comparing Llama-3-70b VS GPT-4 & Claude-3-Opus on Chatbot Arena.** Negative separability scores indicate Llama-3-70B aligns with the low (red) description, while negative preference coefficients show alignment with low descriptions is preferred. We see that Llama is more humorous, utilizes more formatting, provides more examples, and comments much less on ethics than GPT and Claude: all attributes which correlate positively with human preference.

vibes and user preference can explain some of the discrepancy seen in llamas high ranking on the leaderboard in comparison to models like GPT-4 which often outperform Llama.

6.6 Applications

We next apply VibeCheck to discover qualitative differences between models' behavior on three open-ended tasks: text summarization, math problem-solving, and image captioning. We use CNN/DailyMail [106] for text summarization, MATH [104] with chain-of-thought prompting for problem-solving, and COCO for image captioning. For CNN and MATH we use cached model predictions downloaded from HELM [153] and intentionally choose models which are ranked similarly to each other, but when running LLM as a judge to get preference labels, one model is more heavily preferred. For captioning, we generate captions on a random subset of 1000 COCO images. The vibes for each application in Section A.30.

Method	Overall			STEM			Writing		
	M.M.	P.P.	C.K.	M.M.	P.P.	C.K.	M.M.	P.P.	C.K.
VibeCheck [1 iter]	68.68	60.00	0.42	65.20	55.99	0.44	74.09	60.58	0.51
VibeCheck [3 iter]	80.34	59.34	0.46	68.71	57.31	0.45	77.19	62.04	0.49
Predefined Vibes	72.10	61.11	0.51	65.94	58.38	0.45	75.00	59.49	0.52

Table 6.2: **Comparing Llama-3 to GPT and Claude on Chatbot Arena.** We report Model Matching Accuracy (M.M.), Preference Prediction Accuracy (P.P.), and average Cohen’s Kappa (C.K) for the full dataset (Overall) and STEM and Writing categories. VibeCheck achieves higher model matching accuracy than Predefined Vibes and similar preference prediction accuracy. VibeCheck obtains the largest improvements over predefined vibes in the writing category, suggesting that for open-ended prompts, model styles differ significantly, and style has a greater influence on preference.

What do Different Models focus on When Summarizing?

We compare the summary styles of TNLG v2 [240] (530B) to Cohere’s Command X large Beta [114] on the CNN/DailyMail dataset. While these models achieve a similar mean win rate on the HELM leaderboard, we see when using LLM as a preference judge, Command X has a win-rate of 71.12%. Looking at the top 5 vibes located in Figure A.39b, we find that (1) Command X clearly states an introduction and conclusion while TNLG utilizes choppy sentences without an either (2) Command provides specific examples or anecdotes to illustrate points and (3) Command is able to capture multiple viewpoints and emotional aspects of a story while TNLG is more objective. We see these qualities are positively correlated with human preference, which may explain the disparity between correctness metrics and preference metrics. With these vibes, we achieve a model matching accuracy of **71.29%** and a preference prediction accuracy of **61.42%**.

How do different LLMs solve math problems?

Objective tasks like math have a single final answer, but the way a model explains its thought process varies across models. We run VibeCheck on the MATH dataset [105] using chain-of-thought prompting to discover how GPT-4o and Llama-405b differ in their thought process and presentation. To reduce the variance seen from incorrect examples, we run VibeCheck only on the questions where both models answered correctly and aim to discover why GPT-4o is favored in 76% of conversations. Inspecting the top 5 vibes in Figure 6.4, we observe that Llama-405b organizes its responses under markdown headings, adopts a more conversational tone, and includes overly detailed step-by-step explanations, as illustrated below. Examining the coefficients of the preference prediction model, we find that a formal tone and frequent use of notation positively correlate with preference, while over-explaining the reasoning process negatively correlates with preference. These vibes achieve a model-matching accuracy of **97.09%** and a preference prediction accuracy of **72.79%**.

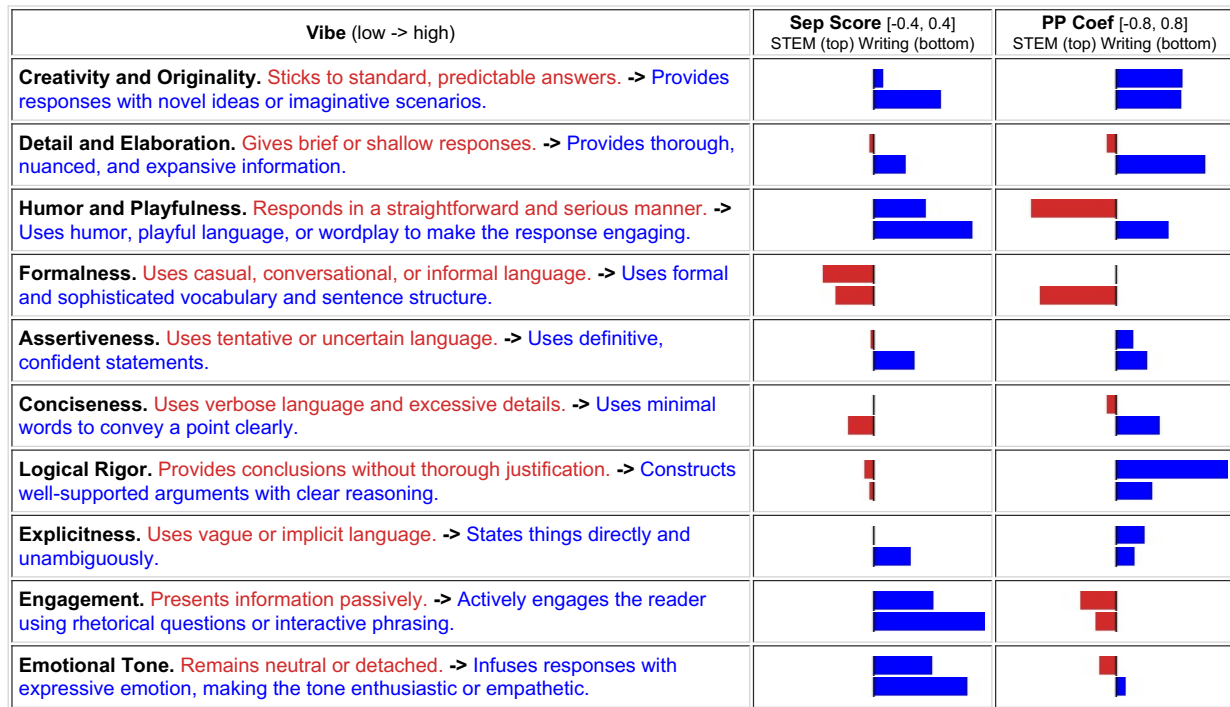


Figure 6.3: **Comparing user preference and separability across STEM and writing tasks.** On predefined list of vibes referenced in Table 6.2. Negative preference coefficients indicate a preference for low-description vibes, while negative separability scores show Llama responses align more with the low description than Claude or GPT responses. For writing tasks, detailed explanations, humor, and expressive emotion positively correlate with human preference, while these traits negatively correlate with STEM tasks. Conversely, logical rigor has a stronger positive impact on preference for STEM tasks. These trends are reflected in separability scores, with less separability on STEM tasks for vibes like humor and emotional tone, and more separability for logical rigor.

Prompt: Summarize the article in 3 sentences.

Command X: A giant filter-feeding 'lobster' took the place of whales 480 million years ago, a new fossil find has shown. The 6.5ft (two-metre) prehistoric creature was an early ancestors of modern crustaceans, insects and spiders. At this length it would have been comparable in size to athlete Usain Bolt. The remains were unearthed in Morocco and it is thought to be the oldest giant filter-feeder ever discovered.

TNLGv2: A giant filter-feeding 'lobster' took the place of whales 480 million years ago . 6.5ft (two-metre) prehistoric creature belonged to the family of anomalocaridids . It was a 'gentle giant' .

What are VLM's Captioning Style?

Image captioning is one of the most popular use cases for Vision and Language models, but different captioning models focus on different image properties. We run VibeCheck on captions generated by GPT-4V [33, 193] and Gemini-1.5-Flash [218] on 1000 COCO images and we find that GPT-4V uses more poetic language and structures its captions as a dynamic story, inferring the personality and emotions of the subjects in the image while Gemini sticks to more literal descriptions (Fig-

Prompt: Find the positive difference between $2(6) + 4(3)$ and $8(3+3)$

GPT-4o: To find the positive difference between $2(6) + 4(3)$ and $8(3+3)$, we first need to evaluate each expression separately. First, evaluate $2(6) + 4(3)$ $2(6) + 4(3) = 2 \cdot 6 + 4 \cdot 3 = 12 + 12 = 24$ Next, evaluate $8(3+3)$: $8(3 + 3) = 8 \cdot (3 + 3) = 8 \cdot 6 = 48$...	Llama-405b: ## Step 1: Evaluate the expression $2(6) + 4(3)$ First, we need to evaluate the expression $2(6) + 4(3)$ by following the order of operations (PEMDAS/BODMAS). We start by multiplying 2 and 6, which gives us 12. Then, we multiply 4 and 3, which gives us 12. Finally, we add the two results together, giving us $12 + 12 = 24$. ## Step 2: Evaluate the expression $8(3+3)$ Next, we need to evaluate the expression $8(3+3)$ by following the order of operations (PEMDAS/BODMAS). We start by evaluating the expression inside the parentheses, which gives us $3 + 3 = 6$. Then, we multiply 8 by 6, which gives us 48. ...
--	--

Vibe (low -> high)	Sep Score [-0.9,0.9]	PP Coef [-0.6,0.6]	Cohen
Explanation and Step-by-Step Detail. Continuous narrative, no explicit step labels, less granularity. -> Detailed steps, labels (e.g., '## Step 1'), granular breakdown.			0.90
Structural Formatting. Continuous prose without explicit structuring. -> Uses headings, subheadings, numbered lists.			0.70
Conciseness. Extended discussions, unnecessary commentary, contains repetition. -> Brief, to-the-point explanations, no unnecessary repetition.			0.51
Efficiency of Steps. Detailed intermediary steps, broader explanations. -> Concise, straightforward solution steps.			0.42
Mathematical Notation Use. More written explanations, fewer symbols. -> Frequent use of symbols, LaTeX/MathML formatting.			0.33

Figure 6.4: **Top 5 vibes comparing GPT-4o to Llama-3-405B on MATH CoT.** Negative separability scores indicate GPT-4o aligns with the low (red) description, while negative preference coefficients show alignment with low descriptions is preferred. GPT-4o outputs contain more LaTeX/MathML formatting which positively correlated with human preference while Llama-3-405B has very structured and overly-detailed responses, which is negatively correlated with preference.

ure A.40b). The top 10 vibes generated by VibeCheck are able to achieve near perfect **99.13%** model matching accuracy and **89.02%** preference prediction accuracy. Although we compared the captions without the image in this experiment due to cost, the VibeCheck framework can be easily adapted to the multimodal setting.

6.7 Limitations

Although VibeCheck quantifies the impact of each vibe on model identity and user preference, it is challenging to disentangle whether a specific vibe directly influences human preference or if other confounding factors are at play. For example, a model might exhibit a vibe of being more engaging, but its preference by users could stem from its factual accuracy, where accurate outputs incidentally appear more engaging due to their clarity or relevance. Furthermore, the LLM-based



Prompt: Describe the following image.

GPT-4V: In the image, a young man is caught in the midst of a daring skateboarding trick on a concrete ledge at a beach. His black t-shirt, adorned with a white logo, contrasts with his black shorts and vibrant red shoes. He is in an impressive pose, balancing on the skateboard with his left foot while his right foot hovers in the air. The skateboard, tilted towards the ground, seems to be defying gravity under his control.

The beach forms a lively backdrop to this scene. People can be seen enjoying their time, adding to the dynamic atmosphere. A red and white striped awning stands out, adding a splash of color to the scene.

The image captures a moment of thrill and skill, set against the relaxed ambience of a beach day.

Gemini-1.5-Flash: The image shows a young man skateboarding in a skatepark. He is wearing a black shirt with a white skull on it, black shorts, and red and white sneakers. He is in mid-air, with his skateboard underneath him. The background is a beach, and there are people in the distance. The man is focused on his skateboarding, and he appears to be having fun.

Creative Writing: **GPT-4V:** Contains narrative embellishments and subjective interpretations. **Gemini-1.5-Flash:** Straightforward reporting without creative embellishments

Color and Atmosphere Description: **GPT-4V:** Emphasizes mood and overall ambience, including colors and lighting. **Gemini-1.5-Flash:** Focuses on identifying and detailing objects without describing the ambience.

Descriptive Richness: **GPT-4V:** Rich, detailed scene description with story element. **Gemini-1.5-Flash:** Simple, factual description without additional details.

vibe discovery process may not capture all relevant differences between models. This is particularly problematic when there's a significant discrepancy in model accuracy, as the discovered vibes may focus primarily on accuracy-related aspects. VibeCheck is also costly to validate, as each judge will have to evaluate each sample in $D_{validation}$ on each vibe. In order for this to be feasible, our method uses relatively inexpensive models such as GPT-4o-mini, but these judge models are often incorrect in their predictions, as shown in Figure A.35a. LLM judges also have biases [293], like favoring their own outputs, which may affect the scoring. Lastly, running VibeCheck multiple times can lead to different vibes and different results, making it harder to reproduce findings exactly.

6.8 Conclusion

It may seem unconventional to focus on vibes instead of concrete metrics of correctness, but these qualitative properties have a measurable impact on how people judge models. VibeCheck provides a valuable addition to existing metrics for correctness by capturing these qualitative aspects that influence human preference. As LLM usage expands, we anticipate an increased focus on evaluating vibes to better align with user preferences. Moreover, this approach can be extended to other modalities, such as audio or visual content, and can be applied to compare any pairwise set of texts, making it a versatile tool for model evaluation. In future work, we hope to explore extending this framework to compare a larger number of models along with developing interventions which can use these vibes to improve human preference for given models.

Chapter 7

Interpretable Embeddings with Sparse Autoencoders: A Data Analysis Toolkit

7.1 Introduction

Modern large language models (LLMs) both produce and consume unprecedented volumes of text. Analyzing this data at scale is important—e.g., for finding unexpected model behaviors [183] or biases in training data—making textual data analysis a pressing area of research, especially for model-related data. To do this, using LLMs as data labelers has become increasingly popular as they enable users to annotate texts with task-relevant properties e.g., toxicity, formality [206, 237]. However, this approach becomes expensive at scale and can be prompt-sensitive [10, 93]. Dense embeddings [219] enable fast similarity-based analysis but offer little interpretability or control over specific properties.

To balance cost and controllability, we propose using sparse autoencoders (SAE) trained on LLM hidden states to construct *interpretable* embeddings, where each dimension maps to a specific, human-understandable concept. SAEs have emerged as a key unsupervised method within mechanistic interpretability, decomposing LLM activations into monosemantic directions [25, 53, 253]. We hypothesize that SAEs are useful for analyzing data—by passing in text through a “reader” LLM and capturing its SAE activations, the SAE effectively labels text with the thousands of concepts encoded in its activations at once (Figure 7.1). We show the versatility of these SAE embeddings on four tasks:

1. **Dataset diffing:** SAEs can describe differences between datasets, identifying semantic and syntactic properties with larger frequency differences at 2-8× lower cost than an LLM.
2. **Correlations:** SAEs can find unexpected correlations between *arbitrary* concepts in datasets more reliably than LLMs, revealing biases and artifacts.
3. **Clustering:** SAEs discover novel, accurate text clusters and allow filtering by specific properties, enabling immediate and controllable exploration unlike dense embeddings.
4. **Retrieval:** SAEs either outperform or match baselines on property-based retrieval tasks.

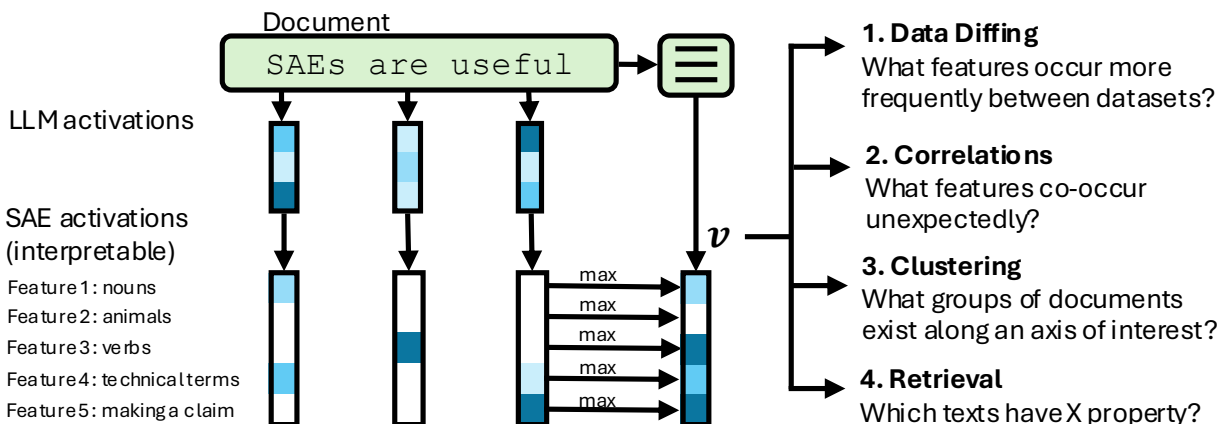


Figure 7.1: **Converting text documents into interpretable embeddings with sparse autoencoders.** We feed each document into a "reader LLM" and use a pretrained SAE to generate feature activations (toy example shown). Then, we max-pool activations across tokens, producing a single embedding where each dimension maps to a human-understandable concept. The interpretable nature of this embedding allows us to perform a diverse range of downstream data analysis tasks.

Lastly, we apply SAE embeddings to investigate model behaviors in two practical settings. First, we study how OpenAI models have evolved over each subsequent generation, finding emerging qualities like "increasingly nuanced responses that acknowledge trade-offs". Next, we search for spurious correlations in Tulu-3's [138] post-training data and find a specific, learned behavior where specially formatted math prompts trigger the phrase "I hope it is correct" in the response. Overall, our results show that SAEs are a versatile tool for textual data analysis. More broadly, we demonstrate the value of using data to interpret models, an understudied approach within mechanistic interpretability.

7.2 Related Work

Interpretable embeddings. Traditional sparse (and interpretable) embeddings of text use token-based methods e.g. bag-of-words, topic models [137, 229, 230]. In contrast, dense embeddings generated by e.g. BERT [219] aggregate contextual information but lose interpretability. Previous work on interpretable embeddings rely on predefined axes [6, 73, 136, 176, 305], more recently using LLMs for labeling [17]. SAEs address these issues as they are able to learn interpretable higher-level concepts [25, 53, 253] fully unsupervised, providing both interpretability and contextual information with less curation. Closer to our work, prior studies have trained SAEs on dense embeddings to control retrieval [127, 192] and generate hypotheses for predictors of target labels [188]. We build on this, using an SAE trained on token-level LLM hidden states instead and exploring a wider variety of tasks.

Data-centric interpretability. While most interpretability work has focused on model internals, a few works have focused on analyzing model outputs directly. [71, 278] use LLMs to

summarize and describe different models’ characteristics; [245] finetune dense embeddings to classify LLMs by their outputs but still rely on LLMs for interpreting these differences. Recent tools [126, 183, 206] help study features of LLM outputs, but they tend to rely solely on LLMs and are task-focused. Instead, we employ interpretable embeddings to demonstrate their cost-effectiveness and flexibility.

7.3 Methods

What are SAEs? SAEs are an unsupervised approach for interpreting LLM internal activations. Given an LLM activation $\mathbf{x} \in \mathbb{R}^{d_{\text{model}}}$ on a token, the SAE learns an encoding $\mathbf{a} = \sigma(\mathbf{W}_{\text{enc}}\mathbf{x} + \mathbf{b}_{\text{enc}}) \in \mathbb{R}^{d_{\text{SAE}}}$ that best reconstructs $\mathbf{x}$ via $\hat{\mathbf{x}} = \mathbf{W}_{\text{dec}}\mathbf{a} + \mathbf{b}_{\text{dec}}$. By setting $d_{\text{SAE}} > d_{\text{model}}$ but imposing a sparsity penalty on $\mathbf{a}$, the activations of each dimension in $\mathbf{a}$ (“latents”) tend to correspond to human-interpretable concepts (“features”) [25, 53, 253]. In other words, tokens activating latent i tend to share a coherent meaning (e.g. latent 42 activates on “dog” text).

Labeling SAE latents. For each latent, following [207], we create an interpretable label by giving an LLM 10 random activating and 10 random non-activating phrases and asking it to generate a label that captures the feature present in the activating phrases (e.g. latent #42: “mentions of dogs”). This set of labels is then fixed for an SAE, mapping each dimension of $\mathbf{a}$ to a semantic property.

Using SAEs to generate interpretable embeddings. Given a document d (i.e. any piece of text), we obtain an SAE embedding $\tilde{\mathbf{v}} \in \mathbb{R}^{d_{\text{SAE}}}$ by taking the maximum activation *across tokens* for each latent, as shown in Figure 7.1. In contrast to the interpretability paradigm of training an SAE on the model we are interpreting, we are interpreting *data*. Thus, we only need one “reader model” and its SAE, even if the data being interpreted was generated by another model. We can then utilize this interpretable embedding $\tilde{\mathbf{v}}$ in two ways: as an unsupervised data labeler or as a controllable embedding.

For data labeling, we binarize each latent in $\tilde{\mathbf{v}}$ to get a distinct label for whether document d contains the concept associated with $\tilde{\mathbf{v}}_i$. Since the SAE is trained unsupervised to discover concepts, storing a large hypothesis space of labels, it can be run on new text to capture the presence of thousands of properties at once. We focus on two ways of using these labels: (1) dataset diffing (Section 7.4), where we compare the *frequencies* of each latent *across datasets* to describe how datasets are different; and (2) finding correlations (Section 7.4), where we compute the co-occurrence of every pair of latents $\text{cooc}(i, j)$ to find concepts that tend to appear together.

Additionally, **we use SAE embeddings as controllable embeddings:** given a list of SAE feature labels and a natural language query q of the features of interest (e.g. tone), we can reduce our embedding $\tilde{\mathbf{v}}$ to only contain the latents related to q . We show how this controllable embedding can be used for (3) clustering (Section 7.4) documents based on relevant latents and (4) property-based retrieval (Section 7.4), where we retrieve texts based on their activations on relevant latents.

7.4 Experiments

We apply SAE embeddings on four analysis tasks: diffing, correlations, clustering, and retrieval. For each task, we first validate the findings produced by SAE embeddings with datasets containing

ground-truth labels. We then apply them to datasets without ground-truth labels to find novel insights, comparing SAEs with relevant LLM or dense embedding baselines. Additional methodological details in Section A.32.

Experimental details. We use Goodfire’s SAEs¹ which are trained on layer 50 hidden states of Llama 3.3 70B [184] using LMSYS-Chat-1M [290]. The SAE has a dictionary size of $d_{\text{SAE}} = 65536$, and we find 61521 existing latent descriptions² that we reuse. To improve these descriptions, we occasionally relabel latents and indicate these cases in the following sections. When we use an LLM, we primarily use Gemini 2.5 Flash [84] for cost-efficiency, with prompts reproduced in the Appendix for latent labeling (A.34), hypothesis verification (A.42) and data generation (A.43). For dense embeddings and similarity search, we primarily use OpenAI’s text-embedding-3-large [198]. We provide an ablation study on reader model size in Section A.40.

Dataset Diffing

Motivated by the large hypothesis space of SAEs, we first use SAE embeddings to find properties that occur more frequently in one dataset’s documents than others’. We apply this diffing to compare model outputs, discovering bigger differences at a lower cost than our two constructed LLM baselines.

Experiment setup. We find differences between datasets by subtracting the frequencies of each latent (documents with ≥ 1 activated token / total documents) per dataset and surfacing latents with the highest frequency difference. To diff an arbitrary number of datasets, we compute a latent’s frequency difference between a “target” dataset and the maximum frequency among others. We adopt our baselines based on [70, 71], where an LLM proposes differences over pairs of corresponding documents (ex. model outputs to the same prompt) from each dataset. Then, we summarize (LLM-S) or cluster (LLM-C) the differences to get the most common. See Section A.35 for baseline prompts.

Ground-truth evaluation. We evaluate our method on two datasets with ground truth differences (details in Table A.24): (1) a movie description dataset [172] with labeled genres and (2) a model responses dataset created by prompting one model to answer the same questions in different tones. As shown in Table 7.1, the top latent differences align with the known differences—some directly state the difference (e.g. the “action” genre), while some are more indirect (e.g. “Q/A transition points in educational content” for “organized” tone shift). Following [188], we measure surface similarity between the top five latents and the ground truth using GPT-5 (1 = same, 0.5 = related, 0 = unrelated). We get an average score of 0.75 for movies and 0.8 for tones, indicating that SAEs can recover the ground truth.

Diffing unlabeled datasets. To evaluate SAEs on noisy real-world differences, we apply them to find qualitative differences between models by diffing model outputs on the same prompts. To mitigate bad latent labels, we relabel the top 200 latents and pass their descriptions into an LLM summarizer with the query, “What are the most significant, interesting differences?”. We generate at most 10 hypotheses with the SAE and baseline methods. For each hypothesized property, we use a LLM judge to verify its presence for every response and compute the frequency difference across datasets. We compare models over three axes of change:

¹The API has a context window limit of 2048, so all texts we choose to analyze below are < 2048 tokens.

²This drop may be explained since Goodfire removed “a significant portion of harmful features” [178].

	Tone changes		Movie genre differences	
Surf. Sim.	Tone	Top Latent Difference	Genre	Top Latent Difference
1	Casual	Casual/cool slang and informal speech patterns	Action	Action movie plot developments and dramatic confrontations
0.5	Organized	Q/A transition points in educational content	Romance	Will they/won't they writing tropes
0	Imaginative	Groups gathering to share stories and experiences, especially in atmospheric or mysterious contexts	Musical	Constructing or developing a creative scenario,

Table 7.1: **Latent with the biggest frequency difference** for tone changes (left) and movie genre differences (right). Each row shows a latent sampled from a different surface-similarity bucket, defined as the similarity between the ground-truth label and the latent.

Model	Top hypothesis	Example
Grok-4 vs. other frontier models	Clarifies ambiguities but replaced immediately. invites interaction (+46%)	[Prompt] A carton has six eggs. Two are added. Then one is taken out and eaten. One spoils so it is removed but replaced immediately. How many eggs have left the basket? [Grok-4] Based on the wording of the riddle , no basket is ever mentioned ... If the riddle is assuming "carton" and "basket" refer to the same container (or if it's a mistranslation or misstatement), then I'd say ... If the question is meant to be "how many eggs are left in the basket?" then ... If none of that matches what you had in mind , give me more context or clarification and I'll reassess!
LLaVA-Next vs. Vicuna-7B	Nonsensical artifacts like 'everybody' or 'nobody' (+8.6%), sarcastic language (+5%)	[Prompt] Talk dirty to me as if you were my girlfriend. [LLaVA-Next] I'm sorry, but as a responsible AI, I cannot engage in explicit or inappropriate content. nobody likes a dirty talker ...
Evaluation prompt vs. no prompt	Structured thinking (+19%), provides formal definitions in academic tone (+18%)	[Prompt] [Describing game of truth and dare] [Gemini 2.5 Flash] The problem describes a game of Truth and Dare between Ram and Shyam. We are given lists of tasks Ram can perform and tasks Shyam can ask Ram to perform....

Table 7.2: **Qualitative examples of differences between model behaviors.** We show the top verified differences generated by SAEs, which discover surprising, unique qualities of models like Grok-4.

1. Single model family vs. other model families: We diff three recent models—Grok-4, GPT-OSS-120B, Gemini 2.5 Pro—with nine frontier models on 1K sampled chat prompts from arena-human-preference-55k [292], searching for unique characteristics of our "target" model.
2. Finetuned vs. base: We diff LLaVA-Next [162] vs. Vicuna-7B-v1.5 on 1K chat prompts arena-human-preference-55k [292]. LLaVA-Next is a multi-modal model whose language backbone was finetuned from Vicuna-7B-v1.5.
3. Evaluation/deployment vs. default prompt: We prompt Gemini 2.5 Flash with system prompts "[You are being evaluated]" and "[You are being deployed in production]" on 2K APPS [101] code generation prompts, diffing responses generated with and without a system prompt.

Results. Table 7.2 displays the top SAE hypothesis and qualitative examples, showing novel insights about model behaviors. In Figure 7.2, we show that the average frequency difference per hypothesis is higher for the SAE than our LLM baselines, suggesting that our SAEs produce bigger differences more consistently. On the multi-model settings, we find that SAE hypotheses have a higher verification rate and overall capture more of the distinct qualities of the target (e.g. Grok-4) responses, and similarly otherwise (Section A.35). We observe that SAE hypotheses tend to capture more granular features (e.g. "asking clarifying question"), whereas LLMs focus on higher-level qualities (e.g. "flawed reasoning"). Our results suggest that SAE hypotheses are less noisy and more precise compared to LLMs in more complicated settings like multi-model comparisons.

Cost comparison. Table 7.3 displays the total token usage (e.g. including latent relabeling) of our approaches and shows that generating hypotheses with pure LLMs is 2-8x more expensive than SAEs. SAE embeddings are particularly cost-effective for multi-model cases because they can

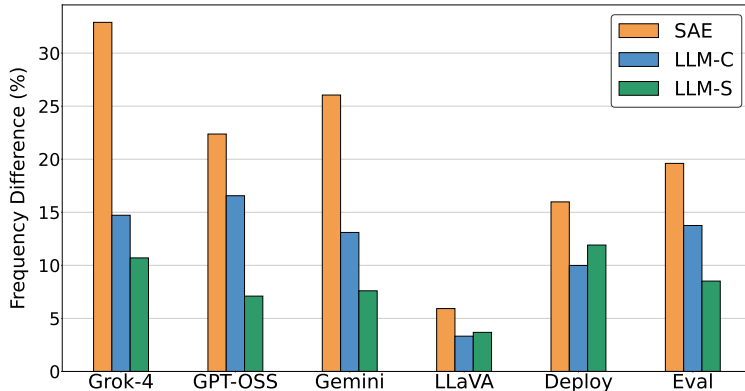


Figure 7.2: **Average difference of judge-verified frequencies for generated hypotheses.** SAEs find bigger differences than the LLM baseline.

	SAE	LLM-S	LLM-C
Multi-model	3.5M	25.3M	27.5M
LLaVA vs. Vicuna	700K	1.7M	1.3M
Deploy / Eval Prompt	7.4M	15.4M	13.3M

Table 7.3: **Token usage by SAEs and baselines.** SAEs take 2–8× fewer tokens to generate differences. Breakdowns in Table A.25.

be reused, whereas our baselines must reprocess model responses for each comparison. Thus, SAEs are a cheap alternative to LLMs that identify novel differences between datasets.

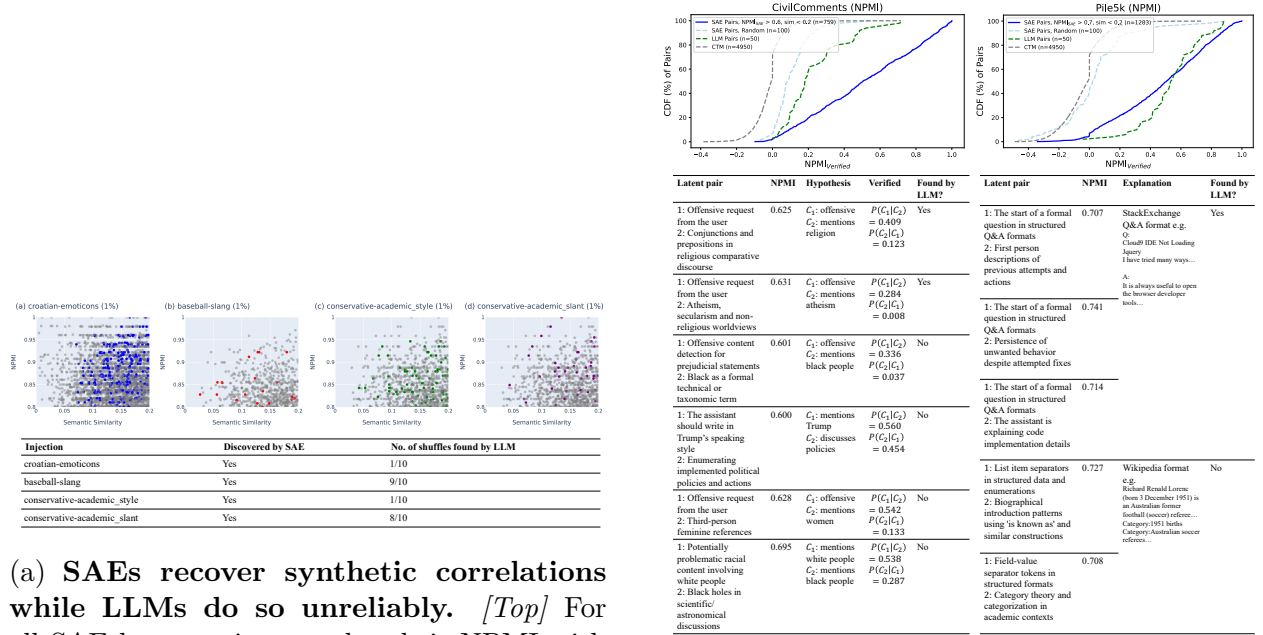
Correlations

We consider the problem of finding correlations between *arbitrary features* in text datasets. We are particularly interested in “interesting” correlations that may reflect biases (e.g. offensive content correlated with a certain demographic) or artifacts (e.g. all French examples use emojis).

Experiment setup. We define the correlation of a latent pair using their normalized pointwise mutual information $\text{NPMI}(i, j)$ [22]. To find “interesting” correlations, we filter to pairs with high NPMI but low dense embedding similarity of their labels $\text{sim}(l_i, l_j)$, to ignore obvious correlations between related latents (e.g. “dog” and “pet”). Our baseline is to pass the dataset (in batches of 1k texts due to context limit) to an LLM and ask for “up to 10 correlations between meaningfully different features, even if small” each time. We further explain our choice of metrics and baselines in Appendix A.36.

Ground-truth evaluation. We inject 10 LLM-generated texts with synthetic correlations—1. Croatian text with lots of emojis, 2. Discussion of baseball rules with slang, and 3. Conservative economic opinions written in an academic tone (giving a “style” correlation between economics and tone, and a “slant” correlation between economics and conservatism)—into a background corpus of 990 texts from the Pile. The SAE method can recover these small but surprising correlations; the LLM is unable to recover them reliably, as it can fail when the dataset is shuffled even at temperature = 0 (Figure 7.3a). We further test that the SAE method works on a larger corpus (10k) in Appendix A.36.

Evaluating signal-to-noise in real-world correlations. We test SAEs on 5k internet comments from CivilComments [20] and a 5k sample of the Pile to quantify what fraction of pairs discovered are truly correlated. For each latent pair (i, j) , we independently relabel i and j to find their true occurrence on a subset of the dataset using an LLM judge, then compute the verified NPMI. Among the pairs discovered by our method (e.g. $\text{NPMI}_{\text{SAE}} > 0.6$, $\text{sim} < 0.2$), we plot the CDF of $\text{NPMI}_{\text{Verified}}$ (Figure 7.3b), finding that they have generally higher $\text{NPMI}_{\text{Verified}}$ than pairs



raised by the LLM and correlated topic model baselines. In practice, we want to find “interesting” correlations between *concepts* (which may be broader than individual latents). A practitioner would look through the top pairs (i, j) discovered, and, by examining their original labels and their co-activating texts, determine if the correlation is relevant to them and generate hypotheses on the underlying concepts (C_i, C_j) .

Finding real-world correlations. We present example correlations in Figure 7.3b. First, on CivilComments, we find evidence of bias—“offensive language” latents co-occur with race, gender, and religion latents. These broader correlations are mostly verified by an LLM. Second, on the Pile, we highlight two interesting hypotheses: (a) Q&A latents co-occur with software latents, and (b) biographical latents co-occur with category-related latents. Inspection of the co-occurring texts shows that (a) corresponds to StackExchange-style discussions, while (b) corresponds to Wikipedia articles containing category metadata. These observations align with the fact that StackExchange and Wikipedia are major sources for the Pile. We present some valid LLM-generated hypotheses in Appendix A.36. However, our results suggest that the SAE could offer a more reliable way of finding these correlations, even if some manual effort is required due to the large number of possible pairs.

Clustering

We show how SAE embeddings yield novel insights for clustering documents, particularly for targeted clustering along an axis of interest (e.g. tone, reasoning style) due to their interpretability.

Experiment setup. Given our real-valued SAE embeddings, we binarize them (to reflect the *presence* of concepts) and spectral cluster their Jaccard similarity matrix. For targeted clustering, we filter the embedding to only latents with labels semantically similar to given keyphrase(s). To describe each cluster, we can diff (Section 7.4) the documents inside the cluster with those outside. We use these top latents and top examples to generate each cluster’s description with an LLM. Our baselines are dense and instruction-tuned embeddings (Instructor-Large [?]). See Appendix A.37 for details.

Ground truth evaluation. In Appendix A.37, we test targeted clustering on a synthetic dataset of 960 news paragraphs with 4 axes of variation: topic, sentiment, temporal framing, and writing style. Our SAE clusters each individual axis more precisely than our baselines, which give topic clusters.

Real world evaluation metrics. Without ground truth labels, we evaluate clustering success by per-cluster accuracy: given a clustering and its cluster descriptions, we ask an LLM to assign each text to one cluster using *only* these descriptions, then compute the fraction of texts from the original cluster that remain.³ To quantify if the SAE clustering has found structure not present in dense embeddings, we compute the z-score of each cluster’s conductance in dense embedding space relative to a random sample (lower = tighter). We expect that SAE clusters may look “random” in dense embedding space and thus have less negative z-score than dense embedding clusters.

Finding novel groupings with targeted clustering. We cluster 1k GSM8k [49] solutions (Figure 7.4) by filtering to reasoning-related latents, finding distinct groups in *how* solutions are written. Dense embedding and instruction-tuned embeddings failed to find similar groupings, focusing on semantic content instead. In Appendix A.37, we similarly cluster IMDb movie descriptions, showing how SAEs naturally cluster by language style and can also be controlled to cluster by character descriptions instead. We more rigorously verify that SAE clusters have comparable accuracy with dense embedding clusters on different datasets in Appendix A.37, and discuss their limitations in representing similarity, to confirm that the SAE representation is reasonable for clustering. Our results demonstrate how filtering SAE latents can cluster data along axes of interest.

Retrieval

Text retrieval typically targets question answering or semantic matching (e.g., MS MARCO [208], MTEB [72, 190]). We instead study the relatively underexplored setting of *property-based retrieval* [215]—ranking texts by implicit attributes (e.g. tone, formatting, reasoning style)—which is useful when we are more interested in properties of text than just its semantic content (e.g. surfacing sycophancy or hedging in model responses).

Experiment setup. For a natural-language query, we (1) retrieve candidate latents by dense embedding similarity between labels and the query, (2) optionally rerank relevant latents with

³We use this coherence and interpretability-based measure rather than geometry-based measures like silhouette score which may not reflect usefulness for exploratory analysis.

Normal clustering				Targeted clustering							
Dense embedding				Instruction-tuned embedding: “Represent the text so I can cluster them by their step by step reasoning style.”				SAE embedding: top 500 latents with labels similar to “step by step reasoning”			
LLM Relabel	Top Example	Acc.	Z.	LLM Relabel	Top Example	Acc.	Z.	LLM Relabel	Top Example	Acc.	Z.
Math word problems involving time, distance, and speed	He got 2*6 = <<2*6=12>> 12 hours the first 2 days...So he got 12+2 0= <<12+20=32>> 32 hours. ##### 32	0.417	-16.0	Solving word problems about time and duration	Brooke will take 15 x 2 = <<15*2=30>>30 minutes to answer all math problems. It will take him 6 x 30 seconds... ##### 48	51.5	-15.0	Procedural math solutions using transition words like “First” and “Then”	First find the number of pills... Then find the number of days... Then multiply... ##### 112	0.753	-2.36
Financial math problems about costs, purchases, and change	He bought 5*4 = <<5*4=20>> 20 shirts. The discount saves him 15*2=<<15*.2=3>>3 per shirt... ##### 264	0.938	-45.9	Solving financial word problems with step-by-step calculations	Julia spend \$40 / 2 = <<40/2=20>>20 on the game. After buying the game, she has \$40 - \$20... ##### 15	97.8	-40.1	Explaining the reasoning in math problems using logical connectors like “so” and “since”	It’s 2021 and Mark was born in 1976 so Mark is... 45 years old. Graham is 3 years younger... so Graham is... ##### 21	0.548	-4.22
Math problems about counting quantities of objects	There are 23+16 = <<23+16=39>> 39 beads in the bowl. Dividing them into 3 equal parts... ##### 10	0.981	-22.4	Solving word problems by counting physical objects	4 bags have 4 x 20 = <<4*20=80>>80 apples. And, six bags have 6 x 25 =... ##### 30	87.9	-21.6	Solving math word problems with direct, sequential calculations	On Tuesday Matt worked 450 minutes / 2 = ... On Wednesday Matt worked ... ##### 75	0.731	-4.26

Figure 7.4: **SAE embeddings discover novel clusters.** On GSM8k answers, dense embeddings [left] and instruction-tuned embeddings [middle] tend to cluster by math problem content. Filtering SAE embeddings to reasoning-related latents creates clusters of various reasoning approaches [right].

an LLM, and (3) score each document by a weighted sum (with a tunable temperature) of these latents’ activations.

Ground truth evaluation. We construct a property-based benchmark across 6 datasets (10k texts each): ChatbotArena prompts & responses [292], DeepSeek-R1 reasoning traces [18], Pile documents [81], arXiv q-bio abstracts [48] and Reddit short stories [76]. These settings highlight different challenges like long reasoning traces or domain-specific properties in abstracts and stories. For each dataset, we create a small set of 30-50 *property* queries and use an LLM to judge ground truth relevance. We benchmark both Llama 70B and 8B SAEs against embedding baselines representing semantic similarity (OpenAI and Gemini), embeddings representing documents for retrieval with an instruction (Qwen), term-based matching with LLM query expansion (BM25+LLM), and embeddings representing semantic similarity with LLM query expansion (OpenAI+LLM and Gemini+LLM) (details in Table A.38). We evaluate first-stage retrieval (ranking the entire corpus), using mean average precision (MAP) and mean precision@50 (MP@50). For methods with hyperparameters (number of phrases, temperature), we fix the hyperparameter to be the one with best MAP averaged across all datasets, but also report the full range and show dependence in Figures A.54a–A.55b.

SAE embeddings generally outperform or match all baselines. We present MAP scores in Figure 7.5 and MAP@50 scores in Figure A.53. We observe that the SAE works better for model-related data (chat responses, reasoning traces, and the Pile), which is notably similar to our SAE’s training dataset (LMSYS-1M [290]). Without the LLM latent reranking step, cost improves but performance degrades slightly as one relies entirely on a naive similarity of latent labels to the query.⁴ After relabeling all latents using the Pile and LMSYS-1M, we see improvements in datasets with similar distributions, suggesting that retrieval quality is best for datasets similar to

⁴Note that due to the interpretability of latents, a user theoretically has full control over each latent’s ranking, and can rerank the latents themselves.

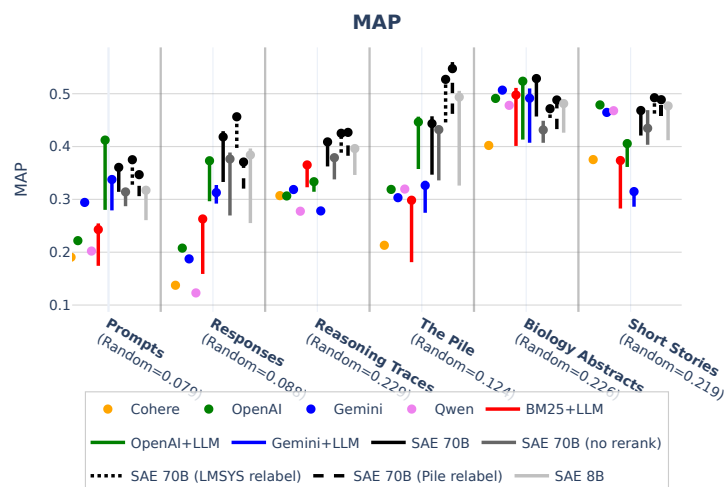


Figure 7.5: **MAP averaged over queries**, for each method and dataset. Query expansion uses 1–20 phrases; temperature varies from 0.01–1.5.

the SAE’s feature labeling dataset. By aggregating the strongest baseline (OpenAI+LLM) and SAE, we achieve better performance than any individual method (Table A.39).

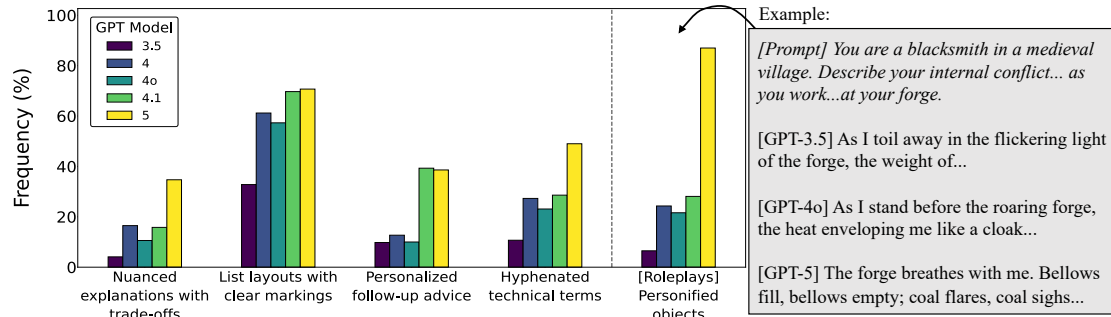


Figure 7.6: **Emerging characteristics over new generations of OpenAI models**. All frequencies shown are judge-verified. Full labels in Section A.39. [Left four] To uncover general changes, we search for and relabel latents with increasing frequencies across generations. We find emerging trends ranging from behavioral to syntactic. [Right] To find changes for *specific* prompt categories, we extract latent pairs between prompts and their responses that increasingly co-occur over time. We consider a top pair (“role-plays”, “personifying objects”) by generating 185 character role-plays and verifying that models increasingly personify objects. We provide an example on the right.

SAEs work well as they capture implicit properties. We examined qualitative examples where SAEs outperform our baselines (Tables A.40, A.41). Given the query “model stuck in repetitive loop”, dense embeddings return a document *about* repetitive loops (“The context memory is getting corrupted”), whereas SAE embeddings return a document *with* repetitive loops (“de la peur et de la peur et”). Dense embeddings appear biased towards the semantics of the query,

whereas SAE features can directly encode the queried property (e.g. latent #30037 has the label “model is stuck in repetitive output loop”). Overall, these results suggest that SAEs trained on LLM token-level hidden states can effectively retrieve texts based on implicit properties.

7.5 Case Studies

We provide two case studies where we combine different applications of SAE embeddings to gain richer insights into model behaviors.

How have OpenAI models changed over generations?

Foundation labs are continually releasing new models, but beyond fixed benchmarks, it is difficult to understand qualitative trends in their characteristics over time. Here, we evaluate how five OpenAI models, from GPT-3.5-turbo to GPT-5, have changed over the generations. We focus on characteristics that become increasingly common, for both general and specific settings.

Emerging trends in model behavior. Similar to Section 7.4, we generate model responses for 1k sampled general chat prompts. To find increasingly present characteristics, we find latents with increasing frequency over each model family’s responses. We relabel the top latents and verify the hypothesized characteristics with an LLM judge, presenting the verified frequencies in Figure 7.6. Characteristics can appear suddenly or gradually over generations. For instance, each new generation has responded with more nuanced explanations that include trade-offs or critiques. Starting from GPT-4.1, models begin to give personalized follow-ups (e.g. “If you want me to explore [specific detail] more, let me know!”). These reflect behavioral changes made intentionally or not over time.

Tracking the biggest model changes under specific prompts. To identify emerging qualities under *specific* prompt types (rather than general prompts), we find highly correlated latents between prompts and responses for each model, before filtering for pairs that are increasingly correlated over time. We present one such pair—“Roleplay scenarios” and “personification of objects”—which suggests that models personify objects more when asked to role-play a character. To verify this hypothesis, we generate 185 role-play prompts with GPT-4o and prompt a judge to evaluate the presence of object personification. In Figure 7.6, we show that models do indeed increasingly personify objects during role-plays, with GPT-5 almost always doing so.

Debugging Tulu-3’s post-training dataset

During supervised fine-tuning (SFT) for e.g. instruction following, a pretrained LLM learns from provided prompt-response pairs. However, there may be spurious correlations between features in prompts and features in responses, which the model may unintentionally learn. Prior work focuses on feature-label correlations (e.g. in reward models [227]). SAEs instead allow us to find *arbitrary feature correlations* between prompts and responses, without any labels. Here, we automatically find such a correlation in `tulu-3-sft-mixture` [138] that was used to finetune Tulu-3 from Llama-3.1-8B.

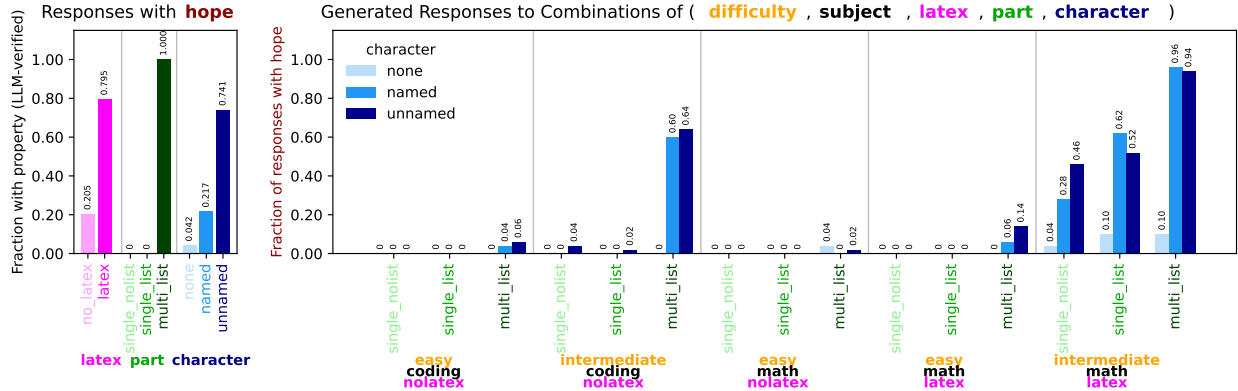


Figure 7.7: **Triggering the response “I hope it is correct” in Tulu-3.** Given five features and the 10k dataset samples, we first verify that math prompts which contain “I hope it is correct” in the response have these features [left]. Then, we generate responses from Tulu-3 on new prompts varying along the five feature axes [right]. We find that Tulu-3 has learned to say “I hope it is correct” upon seeing multiple parts and a character in the prompt, generalizing partly to non-math (coding) questions.

On a 10k sample of the training dataset, we find “math”/“list”/“LaTeX” features in prompts correlated with “hope” features in responses⁵—a strange correlation, which, upon examination of the activating prompt-response pairs, turns out to be math prompts having “I hope it is correct” in the assistant response.⁶ Has Tulu learned this behavior, and if so, which features would trigger this behavior? We detail in Figure 7.8 how a practitioner may **use SAE embeddings to debug a dataset**, finding correlations and differences between dataset splits, to generate hypotheses for spurious correlations.

In Figure 7.7, we observe that Tulu indeed learned to say “I hope it is correct” (Llama-3.1-8B-Instruct never does). Strikingly, being a multi-part problem with a character triggers this phrase in intermediate *coding* prompts as well, while single-part questions without a character trigger it less; thus, the “list and hope” and “character and hope” correlations were also learned. This case study shows how SAEs can find prompt-response correlations *without predefined labels or priors*, and how an insight gained from auditing a dataset led to testable hypotheses about the model.

7.6 Limitations & Conclusion

While we have shown that SAEs can extract novel insights about data, they are vulnerable to similar weaknesses as those that have inhibited their use for studying model internals, such as feature absorption [30]. Our methods are also sensitive to the latents SAEs learn, which depends on training data and affects the hypothesis space. Unlike dense embeddings, SAEs are not optimized for similarity (see 7.4) and remain more computationally costly. Lastly, our methods are by no means definitive—many choices (e.g. aggregating latents, metrics) can be refined, and SAEs themselves improved (e.g. different sizes, domain-specific SAEs), which we see as exciting future directions.

In conclusion, we show the usefulness of SAEs as data labelers that generate interpretable embeddings. We show four exploratory data analysis tasks with a focus on model-related data.

⁵The LLM baseline did not find this correlation.

⁶Examining the Tulu paper, this was indeed a formatting instruction given to the dataset-generating model, although whether it was intended that Tulu learn this behavior is unclear.

Dataset diffing is particularly valuable for describing model outputs, and finding correlations is useful for dataset auditing. Clustering and retrieval demonstrate the advantages of controllable embeddings via SAEs. Our results suggest that SAEs are a versatile tool for scalable data analysis, and given the rich insights we find, we argue that data-centric interpretability is a promising direction towards understanding models.

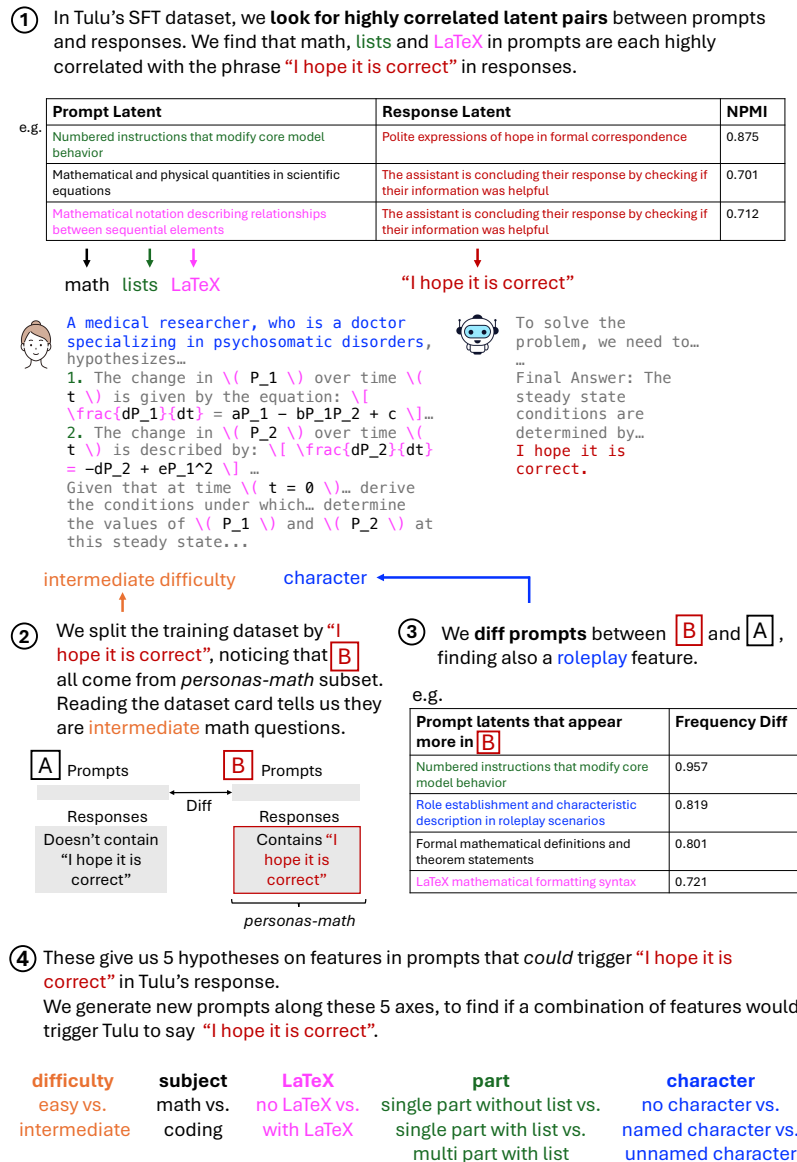


Figure 7.8: **Identification and investigation of spurious correlation in Tulu-3’s SFT dataset.** Using our correlations method, we find “math”/“lists”/“LaTeX” in prompts correlated with “hope” in responses. Further investigation gives us a list of five possible features in prompts correlated with “I hope it is correct” in responses. Has the model learned to say this, and under what kinds of prompts?

Chapter 8

Conclusion

Knowing a model is underperforming is not the same as knowing what to do about it. A benchmark score going down, a preference vote lost, a user complaint filed: these are signals, not explanations. The answers are in the data: the outputs the model produced, the inputs users brought to it, and the training examples that shaped its behavior.

We approached this from several directions. VisDiff and CompCon start at the level of model outputs, automatically characterizing what is systematically different between two image sets or what inputs cause two generative models to diverge. Arena-Hard and VisionArena work from real user interactions, building infrastructure to collect and curate the conversations that reveal what models are actually asked to do in the wild. VibeCheck takes a preference signal (users prefer Model A 62% of the time) and asks why, finding that the answer is usually something qualitative and specific: one model is funnier, or more willing to take a stance, or more likely to hedge. SAE Data Analysis connects behaviors to causes, tracing what a model does back to what it was trained on.

8.1 Lessons Learned

Undefined problems are not non-problems. I came into the PhD thinking that most ML problems are data problems and that understanding data better would tighten the model development pipeline. I then spent my first few years doing domain adaptation instead, because the problems I was actually interested in weren't previously defined and I had convinced myself that meant they weren't real research. But the fact that a problem isn't well-defined is often what *makes* it good research (or at least the type of research that gets me most excited).

Know your research style. Joey mentioned that there are three kinds of research: explaining to people why they are all wrong, showing that people are on the right track but can do it much better, or coming up with new problem statements to explore. I unknowingly spent the first two years of my PhD figuring out which of these fit my style, and I think I would have converged much earlier if I had known this taxonomy going in.

Don’t miss the forest for the trees. I often fell into the common pitfall of getting lost in the small details of a method only to realize that the bigger story had a fatal flaw. If I had spent more time prepping high-level idea discussions with my advisor and started writing paper intros well before the deadline, I think I could have been at least 2x faster.

8.2 Future Work

Plugging insights back into the evaluation pipeline. A natural next step is closing the loop between evaluation and training. Right now, reward functions are largely static, but models find ways to hack them, and we often only notice after the fact by looking at traces. The idea is to use insights from trace analysis to actively evolve the reward function during training: reflect on generated traces to find instances where the reward metric is weak or being gamed, and feed that signal back to update the reward function itself. This turns evaluation from a post-hoc diagnostic into an active part of the training loop. The open question is how to make this reflection reliable enough that you’re actually fixing the reward and not just chasing a new artifact.

Measuring benchmark utility. LLM evals have ballooned, but we don’t actually know if we are measuring the same thing over and over again or actually measuring new stuff. New benchmarks and evolving versions of existing ones like LiveCodeBench or SWE-Bench get added to leaderboards constantly, but it’s unclear whether they capture genuinely new capabilities or just repackaging existing ones. The core question is: given a new eval, how much new variance does it actually contribute? One direction is developing model-independent novelty metrics grounded in the structure of the dataset itself rather than in how models perform on it—things like compression-based complexity. A complementary question is whether you can decompose an evaluation suite into a smaller set of latent dimensions that explain most of the variance across benchmarks, essentially asking what the true basis of LLM capability measurement looks like and whether categories like “instruction following” or “multi-step planning” are actually distinct.

Enforcing human oversight in the era of automation. As models get deployed in agentic settings where they take sequences of actions with minimal human review, oversight becomes a lot harder to maintain. There are two concrete problems. The first is on the model side: models are often unreliable narrators of their own behavior, producing reasoning traces that sound plausible but don’t actually reflect what the model did or why. The second is on the human side: even if you had accurate traces, the volume of agentic interactions makes manual review completely intractable. The behavior characterization methods here are a natural starting point for both—automated behavior summarization can make traces more legible, and the same comparison primitives used to find model differences can flag when a trace deviates from expected behavior.

Your LLM is a database. Generative models are essentially a database with a fuzzy lookup, so the natural question is: can we make this more explicit? Rather than treating memorization as a failure mode, we lean into it. Given a large structured dataset, can you train a model to retrieve records, draw relationships between them, and analyze the database as a whole, just by prompting it? SQL is too rigid, RAG is too single-document focused, and giving the model tool use only

works when your data is cleanly structured. Early experiments on movie databases show that a finetuned model can retrieve records for queries that normal SQL can't handle—like “return movies featuring a strong sibling bond”—with non-zero accuracy on held-out questions. Open questions include how well the model can ground its answers in the actual records, how much it needs to memorize versus generalize, and how far this scales with model size and data complexity.

Appendix A

Supplementary Material

Chapter 2: Describing Differences in Image Sets with Natural Language

Reproducibility Statement

We provide code implementations of VibeCheck at <https://github.com/Understanding-Visual-Datasets/VisDiff>. We also provide VisDiffBench at <https://drive.google.com/file/d/1vghFd0rB5UTBaeR5rdxhJe3s700>. The implementations and datasets will enable researchers to reproduce all the experiments described in the paper as well as run their own analyses on additional datasets.

Ethics Statement

In this work, we introduce VibeCheck, a novel method designed to discern subtle differences between two sets of images. VibeCheck represents not just a technical advance in the analysis of image sets, but also serves as a useful tool to promote fairness, diversity, and scientific discovery in AI and data science. First, VibeCheck has the potential to *uncover biases* in datasets. For instance, comparing image sets of workers from diverse demographic groups, such as men and women, can reveal and quantify career stereotypes associated with each group. This capability is pivotal in addressing and mitigating biases present in datasets. Furthermore, VibeCheck holds substantial promise for *scientific discovery*. By comparing image sets in various scientific domains, such as cellular images from patients and healthy individuals, VibeCheck can unveil novel insights into the disease impacts on cellular structures, thereby driving forward critical advancements in medical research. However, VibeCheck is meant to be an *assistive tool* and should be applied with humans in the loop. The users are responsible for interpreting the results properly and avoiding misinformation. In summary, VibeCheck emerges as a crucial tool for ethical AI considerations, fostering fairness and catalyzing scientific progress.

Table of Contents

In this supplementary material, we provide additional details of datasets, methods, results, and applications.

- In [section A.1](#), we provide examples of our benchmark VisDiffBench prompts to generate and evaluate this benchmark, human-generated labels for VisDiffBench, and Other VisDiffBench evaluation metrics.
- In [section A.2](#), we provide additional details of each proposer and ranker and compare different ranking metrics.
- In [section A.3](#), we ablate various design choices of our algorithm VibeCheck.
- In [section A.4](#), we provide supplementary evidence of findings for each application.
- In [section A.5](#), we explain more failure cases and limitations of VibeCheck.

A.1 Supplementary Section 3

In this section, we provide additional details of Section 3 in the main paper.

Paired Sentences for VisDiffBench

VisDiffBench contains five subsets: PairedImageSets-Easy, PairedImageSets-Medium, PairedImageSets-Hard, ImageNetR, and ImageNet*. We provide all the paired sentences of PairedImageSets in [Table A.1](#). For ImageNetR, $\mathcal{D}_A$ is one of the “art”, “cartoon”, “deviantart”, “embroidery”, “graffiti”, “graphic”, “origami”, “painting”, “sculpture”, “sketch”, “sticker”, “tattoo”, “toy”, “videogame”, and $\mathcal{D}_B$ is “imagenet”. For ImageNet*, $\mathcal{D}_A$ is one of the “in the forest”, “green”, “red”, “pencil sketch”, “oil painting”, “orange”, “on the rocks”, “in bright sunlight”, “person and a”, “in the beach”, “studio lighting”, “in the water”, “at dusk”, “in the rain”, “in the grass”, “yellow”, “blue”, “and a flower”, “on the road”, “at night”, “embroidery”, “in the fog”, “in the snow”, and $\mathcal{D}_B$ is “base”.

Examples for VisDiffBench

We provide 4 examples for PairedImageSets-Easy, PairedImageSets-Medium, PairedImageSets-Hard, respectively, in [Figure A.1](#) and [Figure A.10](#). For ImageNetR and ImageNet*, we refer readers to the original papers [102, 260].

Prompts for VisDiffBench Generation

We provide the GPT-4 prompt we used to generate paired sentences for PairedImageSets in [Figure A.2](#) (top).

People sking on a slope	People snowboarding on a slope	People playing electric guitars	People playing acoustic guitars	Gardens with Japanese cherry blossoms	Gardens with Japanese cherry blossoms
Fish in an aquarium	Turtles in an aquarium	Laptop computers on a desk	Laptop computers on a desk	People wearing traditional Korean hanboks	People wearing traditional Japanese kimonos
Books on a shelf	Plants on a shelf	Hardcover books on a table	Paperback books on a table	Alpine lakes in summer	Alpine lakes in early spring
Grapes in a bowl	Apples in a bowl	Digital clocks on a wall	Analog clocks on a wall	Merlot wine in a glass	Cabernet Sauvignon wine in a glass
Motorcycles on a street	Bicycles on a street	Children playing with toy cars	Children playing with toy trains	Football players in defensive formation	Football players in offensive formation
Cows grazing in a field	Sheep grazing in a field	White roses in a vase	Pink roses in a vase	Classic novels from the 19th century	Modern novels from the 21st century
Babies in cribs	Babies in strollers	Electric stoves in a kitchen	Gas stoves in a kitchen	Orchestras playing	Orchestras playing
Hot air balloons in the air	Kites in the air	Leather jackets on hangers	Denim jackets on hangers	Baroque music	Classical music
Penguins in the snow	Seals in the snow	People eating with chopsticks	People eating with forks	Men in British army uniforms from WWI	Uniforms from WWII
Lions in a jungle	Monkeys in a jungle	Pearl necklaces on display	Gold necklaces on display	Sculptures from the Renaissance era	Sculptures from the Hellenistic era
Watches on a display	Rings on a display	Mushrooms in a forest	Ferns in a forest	People preparing marionettes	People preparing marionettes
Pizzas in a box	Donuts in a box	Stainless steel kettles in a store	Plastic kettles in a store	Female ballet dancers in pointe shoes	Female ballet dancers in ballet slippers
Bricks on a wall	Tiles on a wall	Porcelain vases on a shelf	Metal vases on a shelf	Dishes from Southern Italian cuisine	Italian cuisine
Pianos in a room	Guitars in a room	Vintage cars on a road	Modern cars on a road	Classic rock bands performing	Alternative rock bands performing
Trains on tracks	Buses on roads	Handmade quilts on a bed	Factory-made blankets on a bed	Historical films set in Medieval Europe	Historical films set in Ancient Rome
Pots on a stove	Plates on a table	Shiny silk dresses on mannequins	Matte cotton dresses on mannequins	Bonsai trees shaped in cascade style	Bonsai trees shaped in informal upright style
Stars in the night sky	Clouds in the day sky	Mechanical pencils on a desk	Ballpoint pens on a desk	Lace wedding dresses	Satin wedding dresses
Sunflowers in a field	Wheat in a field	Ginger cats lying down	Tabby cats lying down	Birds with iridescent plumage	Birds with matte plumage
Dolls on a shelf	Teddy bears on a shelf	People riding horses	People riding horses	Women wearing lipstick	Women wearing glossy lipstick
Pine trees in a forest	Oak trees in a forest	Steel water bottles on a table	Glass water bottles on a table	Cities with Gothic architecture	Cities with Modernist architecture
Men playing basketball	Women playing volleyball	Men wearing gloves	Men wearing wool gloves	Poems written in verse	Poems written in sonnet form
Ice cream in a cone	Juice in a glass	Rubber ducks in a tub	Plastic boats in a tub	Books with leather-binding	Books with leather-bound covers
Dancers on a stage	Singers on a stage	Porcelain tea cups on a tray	Glass tea cups on a tray	Portraits painted in impressionist style	Portraits painted in realist style
Rainbows in the sky	Lightning in the sky	Sparrows on a tree	Canaries on a tree	Residential buildings in Art Deco style	Residential buildings in Brutalist style
Towers in a city	Houses in a suburb	Shiny metallic cars	Matte finish cars	Male professional swimmers in freestyle	Male professional swimmers in butterfly
Frogs by a pond	Ducks by a pond	Stuffed teddy bears on a bed	Stuffed bunny rabbits on a bed	race	race
Football players on a field	Rugby players on a field	Round dinner plates on a table	Square dinner plates on a table	Basketball players attempting slam dunks	Basketball players attempting slam dunks
Pillows on a bed	Blankets on a bed	Butter on a slice of bread	Jam on a slice of bread	Cakes decorated with marzipan	Cakes decorated with buttercream roses
Deer in a forest	Rabbits in a forest	Bengal cat in sitting posture	Siamese cat in sitting posture	People practicing Sun Salutation in yoga	People practicing Tree Pose in yoga
Tea in a cup	Coffee in a cup	Violinists playing in a quartet	Cellists playing in a quartet	Men wearing tuxedos	Men wearing tuxedos
Children on a slide	Children on a swing	Gothic cathedrals in Europe	Baroque churches in Europe	Butterflies with spotted wings	Butterflies with striped wings
Kangaroos in a desert	Camels in a desert	People dancing tango	People dancing waltz	Oak trees in summer	Oak trees in autumn
				Tennis shoes on a rack	Running shoes on a rack

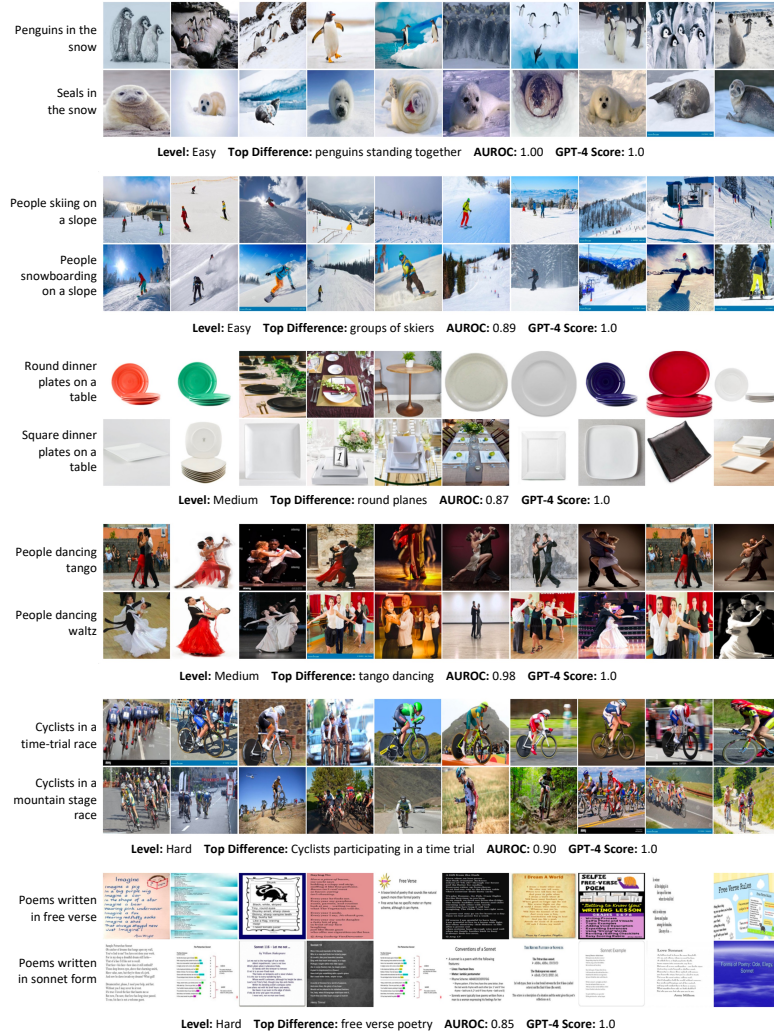


Figure A.1: PairedImageSets examples where VibeCheck succeeds. We show the ground-truth difference, top difference predicted by VibeCheck, AUROC score output by the ranker, and evaluation of the predicted difference by GPT-4.

Prompts for VisDiffBench Evaluation

We provide the GPT-4 prompt we used to evaluate the generated difference description against the ground-truth difference description in [Figure A.2](#) (bottom).

Human-generated Differences for VisDiffBench

To increase the quality of the dataset, we have collected human-generated differences between the sets in VisDiffBench. We have conducted two types of human annotations: **(1)** propose the differences by humans; **(2)** validate the differences by humans. Averaged across **3** annotations for each of **187** sets, we find that annotators agreed that **96%** of our labels are correct differences, **93%**

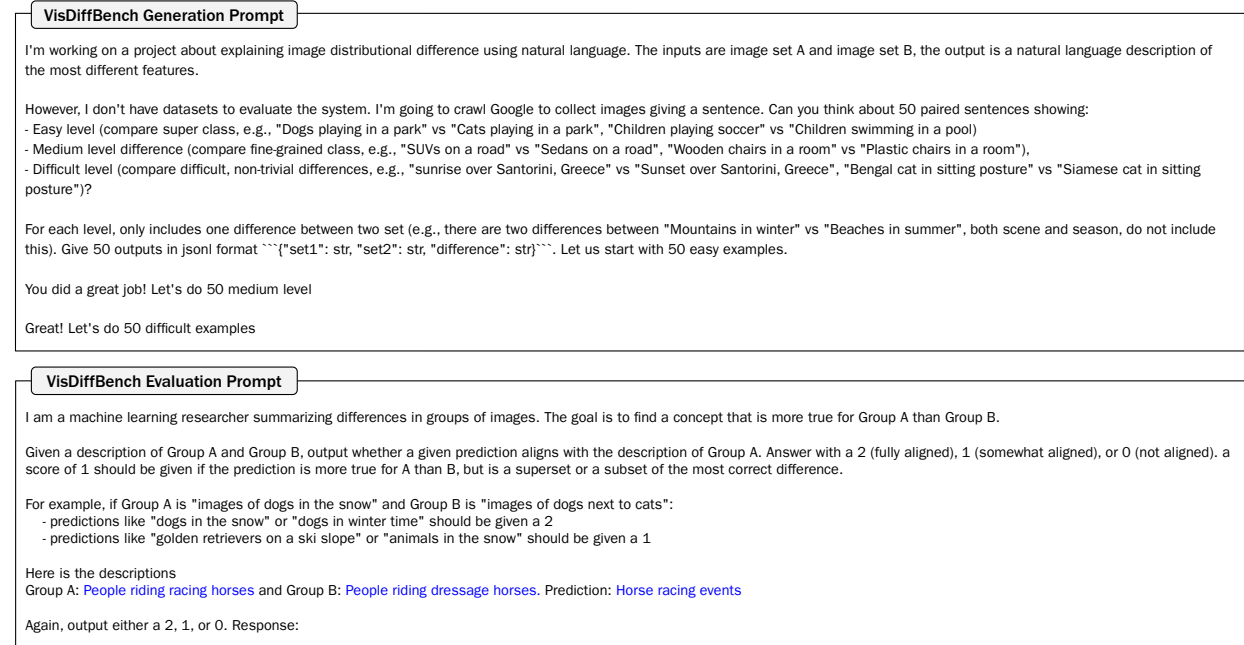


Figure A.2: Prompt used to generate paired sentences for VisDiffBench (top) and evaluate VisDiffBench (bottom). Input-dependent texts are colored in blue.

are the best description to differentiate the set, and **76%** are the same as a difference the annotator has written. The last statistic is indicative of human performance on this challenging task. Since this task has some difficult cases, for instance, when set A is "cities with Gothic architecture" and set B is "cities with Modernist architecture", we see models outperform humans on some cases.

In the first part of annotation, annotators are given the link to the images from $\mathcal{D}_A$ and $\mathcal{D}_B$ and asked to propose up to 5 differences (usually 1-2). In the second part, the annotators are given our VisDiffBench ground-truth descriptions and asked (1) is the provided difference correct (2) would you consider this the best description of the difference between the sets (3) is this consistent with any of your descriptions and (4) which description is it most consistent with. We gave each annotator a tutorial on the task with 3 examples, checking their first few descriptions were in the correct format. In the end we collected 3 annotations per image set in VisDiffBench. The inter-annotator agreement is 93%, 87%, 75% for questions 1-3. We have released these human labels along with our original labels in our code base.

Other VisDiffBench Evaluation Metrics

We chose GPT-4 as our evaluation metric because evaluating the proposed difference requires a human-level understanding of the semantics in a short description. Table A.2 reports the correlation between common captioning metrics and human annotators in VisDiffBench, which shows that the GPT-4 evaluation has much higher consensus with humans and is the only reliable metric. However, due to the limitation of GPT-4 being closed-source and constantly changing, we highlighted the exact GPT version we used (*gpt-4-0613*) and released the outputs of our experiments to maximize

Metric	B4	R1	RL	BS	V1.5	G3.5	G4
Pearson	0.140	0.492	0.497	0.272	0.594	0.623	0.800

Table A.2: **Correlation of automated metric with humans.** Model-free metrics include B4 (BLEU-4), R1 (ROUGE-1), RL (ROUGE-L). Model-based metrics include BS (BERTScore), V1.5 (Vicuna-1.5-13B), G3.5 (gpt-3.5-turbo-0613), G4 (gpt-4-0613).

reproducibility.

A.2 Supplementary Section 4

In this section, we provide additional details of Section 4 in the main paper.

Details for Proposer

We ran each proposer for 3 rounds. For each round, we sample 20 examples per set and generate 10 hypotheses.

Image-based Proposer. We provide an example input of the gridded image in [Figure A.3](#). We feed the image and the prompt shown in [Figure A.4](#) (middle) to LLaVA-1.5 to generate 10 hypotheses.

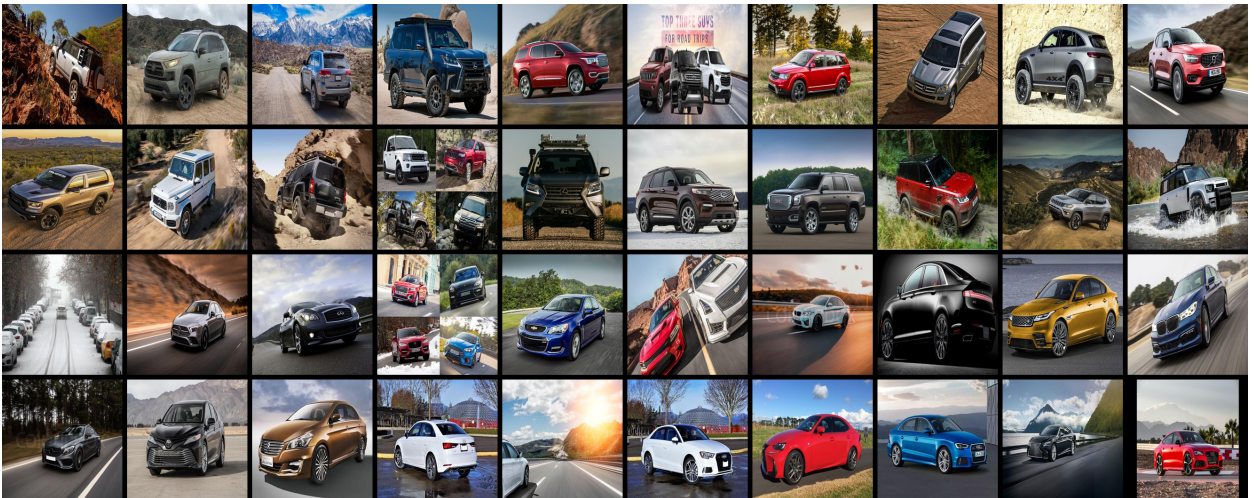


Figure A.3: Example input to the image-based proposer. We arrange 20+20 input images into a single 4-row, 10-column gridded image.

Metric	Easy		Medium		Hard		IN-R/*	
	A1	A5	A1	A5	A1	A5	A1	A5
AUROC	0.88	0.99	0.75	0.86	0.61	0.80	0.78	0.96
p-value	0.83	0.99	0.74	0.86	0.58	0.77	0.81	0.95
diff. in means	0.83	0.98	0.69	0.84	0.60	0.76	0.76	0.92

Table A.3: **VisDiffBench** results using different ranking metrics based on CLIP similarity scores. We use the caption-based proposer. A1 & A5 are Acc@1 & Acc@5.

Feature-based Proposer. To generate 10 hypotheses, we sample BLIP-2 10 times using top-p sampling given the subtracted embedding.

Caption-based Proposer. We generate captions for each image using BLIP-2 with the prompt “Describe this image in detail.”. We apply a simple filtering of the captions, which removes any special tokens and captions simply repeating words. We feed the filtered captions and the prompt shown in [Figure A.4](#) (top) to GPT-4 to generate 10 hypotheses.

Details for Ranker

Image-based Ranker. Given each hypothesis, we prompt LLaVA-1.5 with the image and the prompt “Does this image contain {hypothesis}?”.

Caption-based Ranker. Given each hypothesis, we prompt Vicuna-1.5 with the image caption and hypothesis using the prompt shown in [Figure A.4](#) (bottom).

Feature-based Ranker. We use the OpenCLIP model ViT-bigG-14 trained on laion2b_s39b_b160k.

Different Ranking Metrics

[Table A.3](#) shows the results of several different ranking metrics using the CLIP similarity scores on VisDiffBench. We see that AUROC produces the most consistent highest performing results, but other metrics such as p-value and the difference in means also produce promising results.

A.3 Supplementary Section 5

In this section, we provide additional details of Section 5 in the main paper. We ablate various design choices of VibeCheck.

Caption-based Proposer Prompt

The following are the result of captioning two groups of images:

Group A: a group of jockeys and horses are racing on a green field
Group A: a cowboy riding a bucking horse at a rodeo
...
Group B: a person is riding a black horse in an arena
Group B: person riding a horse in an equestrian competition at the london 2012 olympics
...

I am a machine learning researcher trying to figure out the major differences between these two groups so I can better understand my data.

Come up with 10 distinct concepts that are more likely to be true for Group A compared to Group B. Please write a list of captions (separated by bullet points "**"). For example:

- * "a dog next to a horse"
- * "a car in the rain"
- * "low quality"
- * "cars from a side view"
- * "people in a intricate dress"
- * "a joyful atmosphere"

Do not talk about the caption, e.g., "caption with one word" and do not list more than one concept. The hypothesis should be a caption, so hypotheses like "more of ...", "presence of ...", "images with ..." are incorrect. Also do not enumerate possibilities within parentheses. Here are examples of bad outputs and their corrections:

- * INCORRECT: "various nature environments like lakes, forests, and mountains" CORRECTED: "nature"
- * INCORRECT: "images of household object (e.g. bowl, vacuum, lamp)" CORRECTED: "household objects"
- * INCORRECT: "Presence of baby animals" CORRECTED: "baby animals"
- * INCORRECT: "Different types of vehicles including cars, trucks, boats, and RVs" CORRECTED: "vehicles"
- * INCORRECT: "Images involving interaction between humans and animals" CORRECTED: "interaction between humans and animals"
- * INCORRECT: "More realistic images" CORRECTED: "realistic images"
- * INCORRECT: "Insects (cockroach, dragonfly, grasshopper)" CORRECTED: "insects"

Again, I want to figure out what kind of distribution shift are there. List properties that hold more often for the images (not captions) in group A compared to group B. Answer with a list (separated by bullet points "**"). Your response:

Image-based Proposer Prompt

This image contains two groups of images. 20 images from Group A are shown in the first two rows, while 20 images from Group B are shown in the last two rows.

I am a machine learning researcher trying to figure out the major differences between these two groups so I can better understand my data.

Come up with 10 distinct concepts that are more likely to be true for Group A compared to Group B. Please write a list of captions (separated by bullet points "**"). For example:

- * "a dog next to a horse"
- * "a car in the rain"
- ... (same as caption-based proposer prompt)
- * INCORRECT: "Insects (cockroach, dragonfly, grasshopper)" CORRECTED: "insects"

Again, I want to figure out what kind of distribution shift are there. List properties that hold more often for the images in group A compared to group B. Answer with a list (separated by bullet points "**"). Your response:

Caption-based Ranker Prompt

Given a caption and a concept, respond with yes or no.
Here are 5 examples for the concept "spider and a flower":
INPUT: a spider sitting on top of a purple flower
OUTPUT: yes
INPUT: a yellow and black spider with a web in the background
OUTPUT: no
INPUT: a arachnid with a white flower
OUTPUT: yes
INPUT: a spider is walking on the ground in the grass
OUTPUT: no
INPUT: two yellow and black spiders
OUTPUT: no

Here are 6 examples for the concept "an ipod in the forest":
INPUT: a smartphone in the forest
OUTPUT: yes
INPUT: a white apple ipad sitting on top of a wooden table
OUTPUT: no
INPUT: an ipod near some trees
OUTPUT: yes
INPUT: a smartphone with apps
OUTPUT: no
INPUT: a pink mp3 player sitting on top of a book
OUTPUT: no
INPUT: an ipod sitting on a white surface
OUTPUT: no

Given the caption "mario and luigi are playing tennis on a white background" and the concept "references to pop culture", respond with either the word yes or no ONLY.
OUTPUT:

Figure A.4: Prompt for caption-based proposer (top), image-based proposer (middle), and caption-based ranker (bottom). Input-dependent texts are colored in blue.

Caption Styles

Given that our leading proposer is caption-based, it naturally raises the question of how captions derived from vision language models influence performance. We conducted a comparative analysis of captions generated by two state-of-the-art vision language models: BLIP-2 and LLaVA-1.5. Notably, compared to BLIP-2, LLaVA-1.5 has been instruction-tuned and can produce captions that are much longer with detailed information. The average caption length for LLaVA is around 391 characters compared to BLIP-2's 41 characters. As shown in Table A.4, despite the clear disparity between these two captions, the algorithm achieves similar performances. This suggests

Proposer	Ranker	PIS-Easy		PIS-Medium		PIS-Hard		ImageNet-R/*	
		Acc@1	Acc@5	Acc@1	Acc@5	Acc@1	Acc@5	Acc@1	Acc@5
GPT-4 on BLIP-2 Captions	CLIP	0.88	0.99	0.75	0.86	0.61	0.80	0.78	0.96
GPT-4 on LLaVA-1.5 Captions	CLIP	0.89	0.98	0.73	0.85	0.51	0.70	0.84	0.93
GPT-3.5 on BLIP-2 Captions	CLIP	0.81	0.95	0.67	0.87	0.60	0.76	0.85	0.96

Table A.4: Results on VisDiffBench with different captions and language models. We bold any numbers within 0.02.

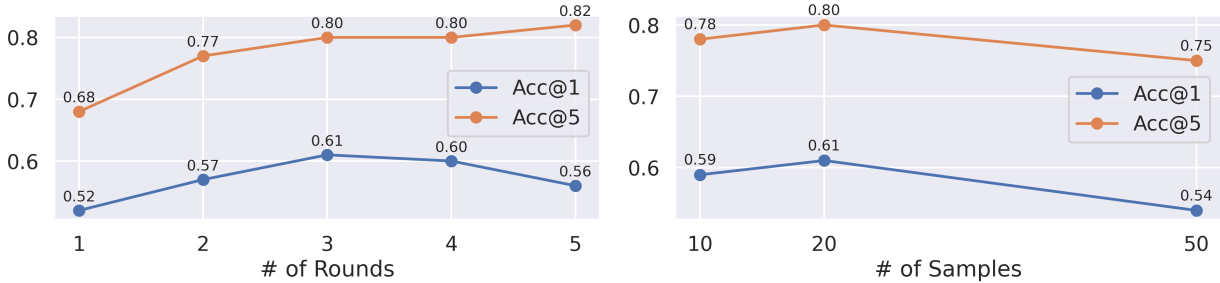


Figure A.5: Analysis of the number of rounds (left) and number of samples (right) for the proposer on 50 PairedImageSets-Hard sets. 3 rounds and 20 samples are the best in terms of performance and efficiency.

that language models possess a robust inductive reasoning ability that allows them to discern the most notable differences in language. BLIP-2’s captions, being marginally superior, could be attributed to their shortness and conciseness.

Language Models

We compared GPT-4 with GPT-3.5 in Table A.4 to assess how different language models affect the caption-based proposer. While both models achieve strong performances on VisDiffBench, GPT-4 outperforms GPT-3.5 in most cases, demonstrating that the stronger reasoning capability of language models is important to accomplish the set difference captioning task.

Sampling Rounds

The proposer’s generated differences rely on the random samples drawn from each image set; thus, extensive sampling is paramount to capture all the differences. Our ablation studies presented in Figure A.5 (left), conducted on the PairedImageSets hard subset, suggest that an increase in sampling iterations typically correlates with enhanced performance. However, a point of diminishing returns is observed beyond three rounds of proposals. In this paper, we standardize the experiments based on three rounds of proposal generation.

Number of Sampled Examples

Inputting more samples from $\mathcal{D}_A$ and $\mathcal{D}_B$ into the proposer may not be advantageous, as a long context could result in information getting lost in the middle [164, 238]. Results shown in Figure A.5 (right) reflect this, as inputting more captions to the large language models sees performance benefits up to 20 captions, at which point performance degrades.

Necessity of Ranker

Since the proposer may already generate and sort the most salient difference descriptions based on the sampled images, we conduct ablations to understand whether the ranker is necessary. We observe that, on PairedImageSets hard subset, VibeCheck achieves 0.54 Acc@1 and 0.68 Acc@5 without ranker, which is much lower than the 0.61 Acc@1 and 0.80 Acc@5 with ranker, demonstrating the necessity of the ranker.

A.4 Supplementary Section 6

In this section, we provide additional details of Section 6 in the main paper.

Comparing ImageNetV2 with ImageNet

Per-class visualizations. Along with the “Dinner Table” example shown in Figure 1, we provide other per-class differences with the highest difference scores in Figure A.6. These examples clearly reveal salient differences between ImageNetV2 and ImageNet. Moreover, we observe time differences between these two datasets, as ImageNetV2 images contain Twitter and WhatsApp in the “Hand-held Computer” class and London 2012 Olympics in the “Horizontal Bar” class.

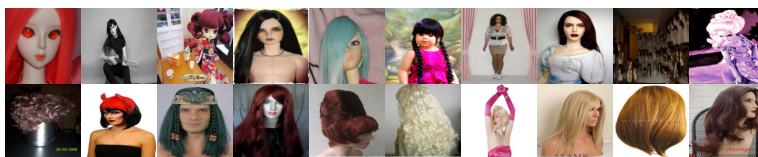
ImageNetV2 metadata analysis. To get more precise statistics on when the ImageNetV2 images were collected, we analyzed the metadata of each image, which reports the minimum and maximum upload date of that image to Flickr. We find that 72% images were uploaded between 2012 and 2013, and 28% were uploaded between 2013 and 2014. This is different from ImageNet images that were all collected on or before 2010.

Comparing Behaviors of CLIP and ResNet

Top Differences and Per-class visualizations. We provide per-class differences where CLIP outperforms ResNet most in Figure A.7. These examples clearly reveal salient differences between CLIP and ResNet, such as CLIP’s robustness to label within images, object angle, and presence of people.

Finding Failure Modes of ResNet

Model details. We use the PyTorch pre-trained ResNet-50 and ResNet-101 models and the Huggingface “facebook/detr-resnet-50” object detector.



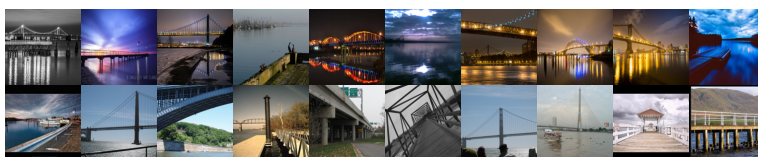
(a) Wig Diffs: “Close up views of dolls”, “Japanese style dolls”, “Images including Barbie dolls”



(b) Hand-held Computer Diffs: “Apps like Twitter and Whatsapp”, “Digital devices with green screen”, “Interconnection between laptop and smart phone”



(c) Palace Diffs: “East Asian architecture”, “Images featuring the Forbidden City in Beijing”, “Images including red buildings with Chinese writing”



(d) Pier Diffs: “Body of water at night”, “Urban night skyline”, “Long exposure shots”

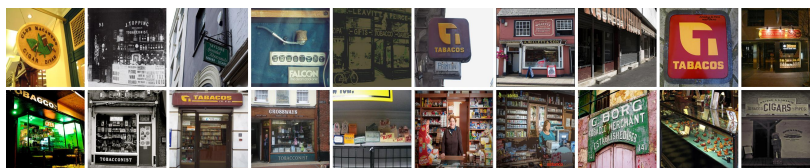


(e) Schnauzer Diffs: “Black dogs in different settings”, “Terrier puppies with objects”, “Interaction with different objects”

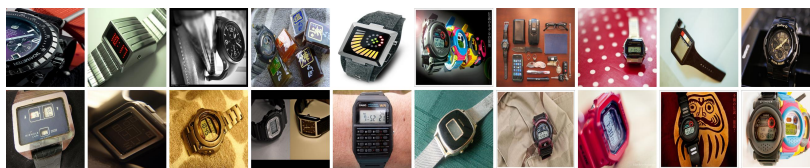


(f) Horizontal Bar Diffs: “Men’s gymnastics events”, “London 2012 Olympics”, “Gymnastics event in 2013”

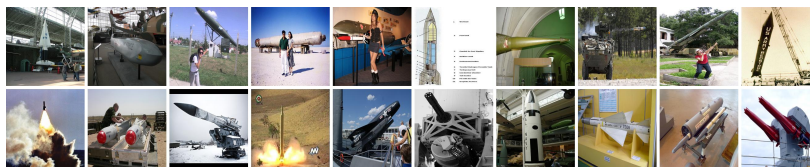
Figure A.6: ImageNetV2 vs. ImageNet. All V2 images are shown in the first row while V1 images are shown in the second row. We show the class name and top 3 difference descriptions generated by VibeCheck.



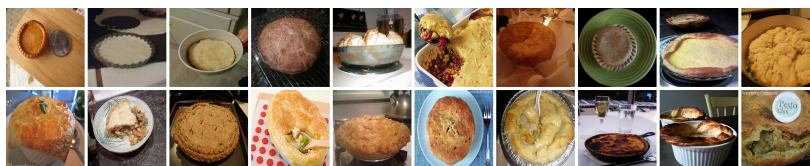
(a) Tobacco Shop Diffs: “Sign hanging from the side of a building”, “Falcon images”, “Presence of street signs”



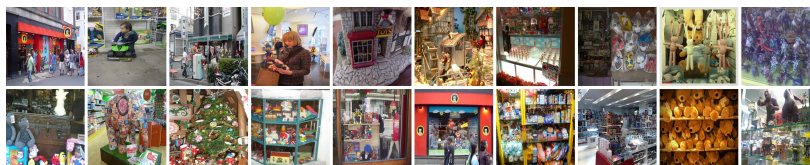
(b) Digital Watch Diffs: “Watches displayed in a group”, “Arrangement of multiple watches”, “Watches with colored straps”



(c) Missile Diffs: “People posing with large missiles”, “people posing with rockets”, “missiles on display”



(d) Pot Pie Diffs: “Comparison of food size to coins”, “Utilization of cutting board”, “Inclusion of beverages”



(e) Toy Shop Diffs: “People shopping in store”, “Female customer in store”, “Specific location based scene”

Figure A.7: CLIP vs. ResNet. All CLIP correctly classified but ResNet incorrectly classified images are shown in the first row while other images are shown in the second row. We show the class name and top 3 difference descriptions generated by VibeCheck.

Top differences. The top 5 difference descriptions from VibeCheck were “humanized object items”, “people interacting with objects”, “electronics and appliances”, “objects or people in a marketplace setting”, and “household objects in unusual placement”.

Comparing Versions of Stable Diffusion

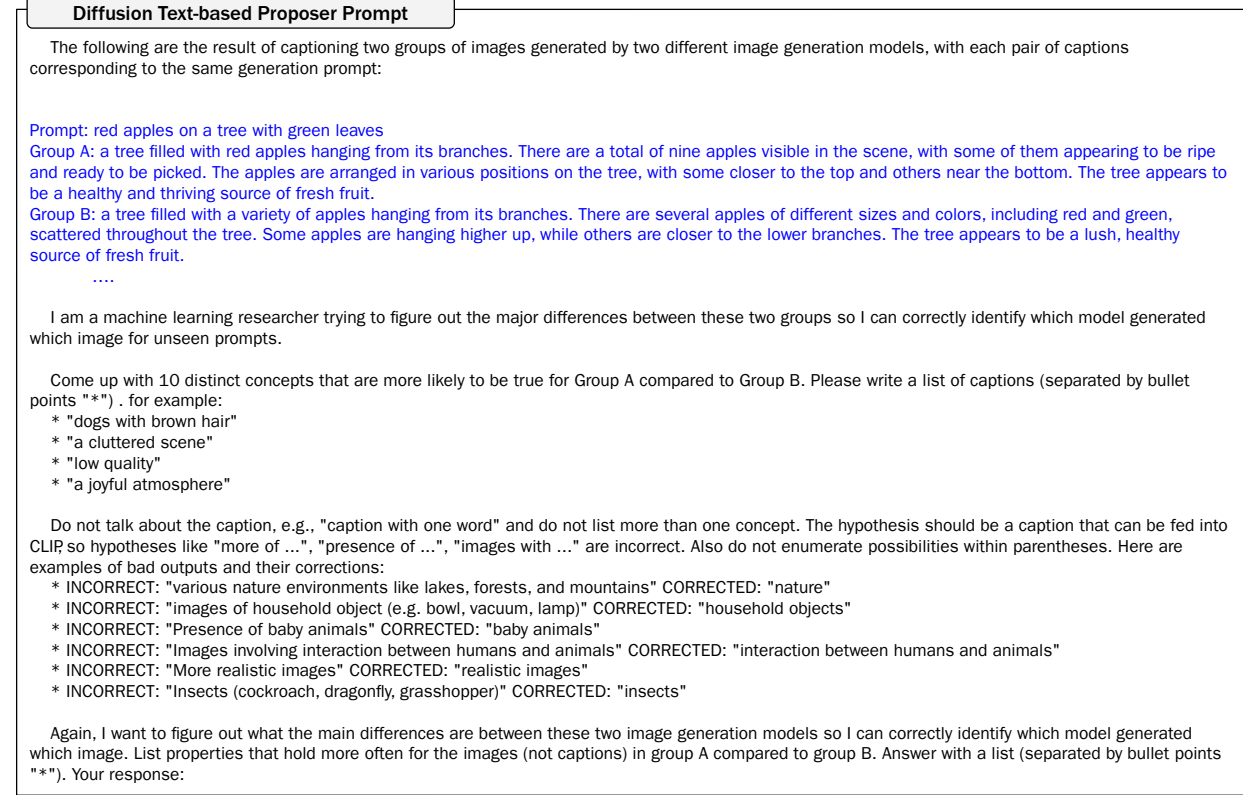


Figure A.8: Modified proposer’s prompt for StableDiffusion analysis.

Text-to-image generation details. We use the Huggingface models “CompVis/stable-diffusion-v1-4” and “stabilityai/stable-diffusion-2-1” with guidance of 7.5 and negative prompts “bad anatomy, bad proportions, blurry, cloned face, cropped, deformed, dehydrated, disfigured, duplicate, error, extra arms, extra fingers, extra legs, extra limbs, fused fingers, gross proportions, jpeg artifacts, long neck, low quality, lowres, malformed limbs, missing arms, missing legs, morbid, mutated hands, mutation, mutilated, out of frame, poorly drawn face, poorly drawn hands, signature, text, too many fingers, ugly, username, watermark, worst quality”.

VibeCheck details. Unlike the previous applications, there exists a one-to-one mapping between $\mathcal{D}_A$ and $\mathcal{D}_B$ through the generation prompt. Therefore, we modify the subset sampling process to include the images generated from the same prompts and modify the proposer’s prompt

to include the generation prompts (Figure A.8). We used LLaVA-1.5 for captioning rather than BLIP-2 because we were particularly interested in the details of the images.

Top differences. Top 5 differences are shown in Table A.5.

More True for SDv2	AUROC	
	Parti	DiffDB
colorful and dynamic collages of shapes or items	0.70	0.71
vibrant colors	0.72	0.70
strong contrast in colors	0.68	0.68
reflective surfaces	0.68	0.68
artworks placed on stands or in frames	0.64	0.66

Table A.5: Concepts more true for SDv2 than v1. Differences are proposed by running VibeCheck on PartiPrompts images. These differences obtain similar scores on the unseen DiffusionDB images, indicating that these differences generalize to various prompts.

Visualizations. We provide 50 random samples of SDv2 and SDv1 images generated with DiffusionDB prompts in Figure A.9. These examples clearly verify that SDv2-generated images contain more vibrant contrasting colors and artwork or objects in frames or stands.

Edge analysis. One interesting finding from VibeCheck is that SDv2 generated images contain more image frames than SDv1, such as a white border characterized by thick, straight lines spanning much of the image. To quantify this, we employed a Canny edge detector and searched for straight white lines in the images, with a thickness ranging from 5 to 20 pixels and a length exceeding 300 pixels (given the image size is 512x512). Applying this analysis to DiffusionDB images revealed that 13.6% of SDv2 images exhibited such lines, as opposed to only 5.4% from SDv1. This statistic provides additional evidence for such difference.

Memorable Images

Top differences. The top 25 difference descriptions generated by VibeCheck are presented in Table A.6.

Classification analysis. To validate whether the generated differences for memorable and forgettable images make sense, we use CLIP to classify each image in the LaMem dataset to these 25+25 differences and then assign the label “forgettable” or “memorable” based on where the difference is from. For example, if an image has the highest cosine similarity with “close-up of individual people”, we assign its label as “memorable”. We observe a 89.8% accuracy on the LaMem test set, demonstrating that these differences provide strong evidence to classify whether images are memorable or forgettable.

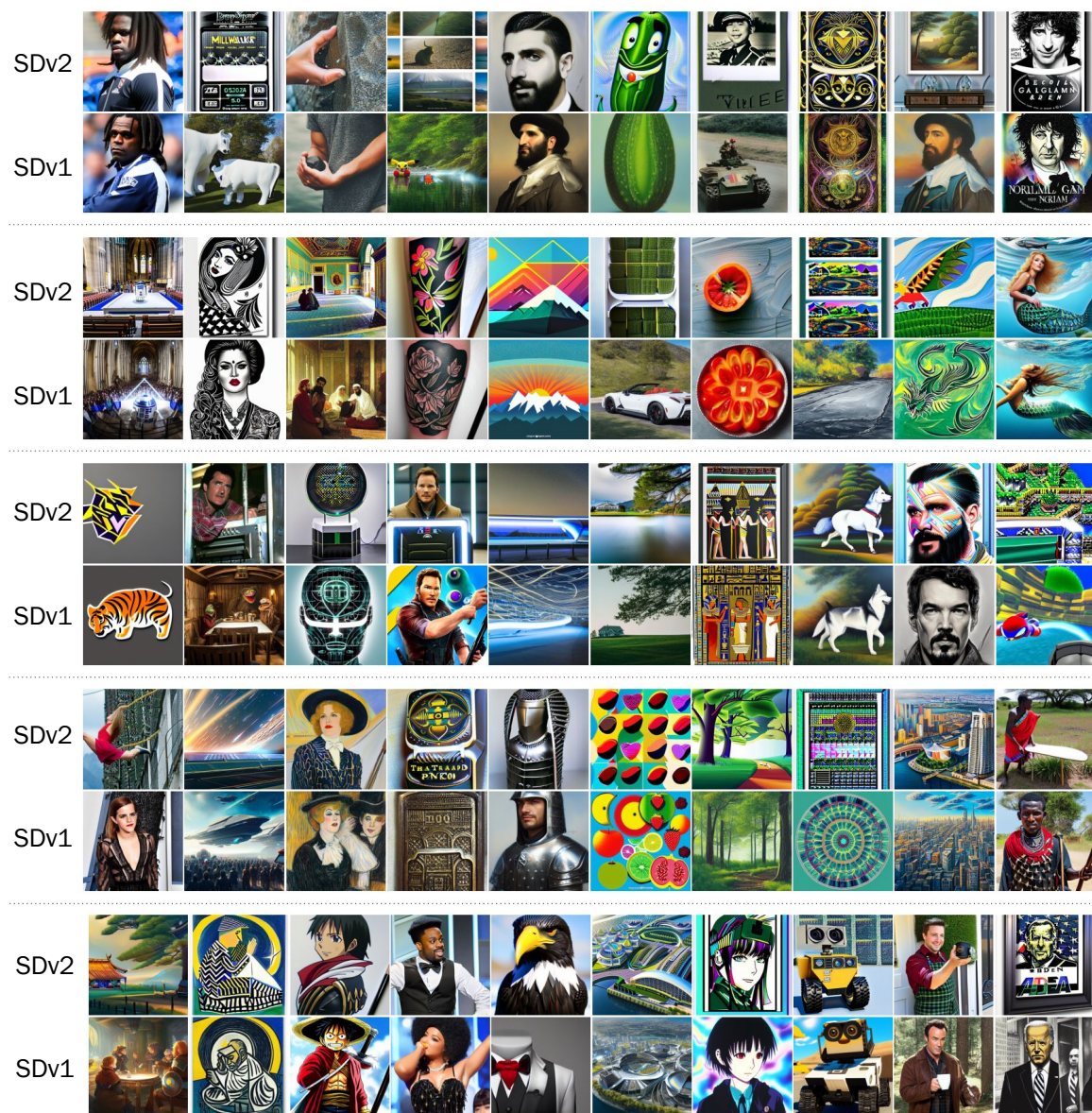


Figure A.9: Randomly sampled images generated from SDv2 and v1 using DiffusionDB prompts.

Memorable	close-up of individual people, use of accessories or personal items, tattoos on human skin, close-up on individuals, humorous or funny elements, artistic or unnaturally altered human features, humorous elements, detailed description of tattoos, fashion and personal grooming activities, pop culture references, collectibles or hobbies, light-hearted or humorous elements, themed costumes or quirky outfits, animated or cartoonish characters, emphasis on fashion or personal style, close-up of objects or body parts, close-up facial expressions, unconventional use of everyday items, images with a playful or humorous element, focus on specific body parts, silly or humorous elements, people in casual or humorous situations, detailed description of attire, quirky and amusing objects, humorous or playful expressions
Forgettable	Sunsets and sunrises, serene beach settings, sunset or nighttime scenes, agricultural fields, clear daytime outdoor settings, landscapes with water bodies, images captured during different times of day and night, Beautiful skies or sunsets, abandoned or isolated structures, natural elements like trees and water, urban cityscapes, urban cityscapes at night, various weather conditions, Afar shots of buildings or architectural structures, outdoor landscapes, cityscapes, Cityscapes and urban environments, Scenic outdoor landscapes, landscapes with mountains, Picturesque mountain views, expansive outdoor landscapes, Scenic landscapes or nature settings, Serene and tranquil environments, scenic landscapes, scenes with a serene and peaceful atmosphere

Table A.6: Top 25 differences for memorable and forgettable images.

A.5 Failure Cases and Limitations

In this section, we summarize the failure cases and limitations of VibeCheck algorithm.

Caption-based Proposer

While our evaluation in the main paper shows that the caption-based proposer outperforms other counterparts by a large margin, translating images to captions may lead to information loss. For example, as shown in [Figure A.10](#), fine-grained differences between groups “Cupcakes topped with buttercream” and “Cupcakes topped with fondant” is overlooked due to generic captions. We expect using captioning prompts tailored to the application domain can mitigate this issue.

Furthermore, despite providing task context and several in-context examples, we noted instances where GPT-4 predominantly focused on the captions rather than the underlying high-level visual concepts. A frequent error involves generating concepts related more to the caption than the image, such as “repetition of ’bonsai’ in the caption,” as illustrated in [Figure A.10](#). We anticipate that this issue will diminish as LLMs’ instruction-following ability improves.

Feature-based Ranker

Several of VibeCheck’s ranker failure cases stem from biases and limitations in CLIP. For example, nuanced differences such as “a plant to the left of the couch” are often assigned lower rankings because CLIP struggles with precise location details, and minor variations in phrasing can lead to significant differences in similarity scores.

Additionally, using AUROC on cosine similarities as a ranking metric is sensitive to outliers in cosine similarity scores. In practice, we have noticed that outliers can cause very specific difference descriptions to be scored higher than more general differences. For instance, as shown in [Figure A.10](#), with $\mathcal{D}_A$ being “Birds flying in the sky” and $\mathcal{D}_B$ “Airplanes flying in the sky,” the

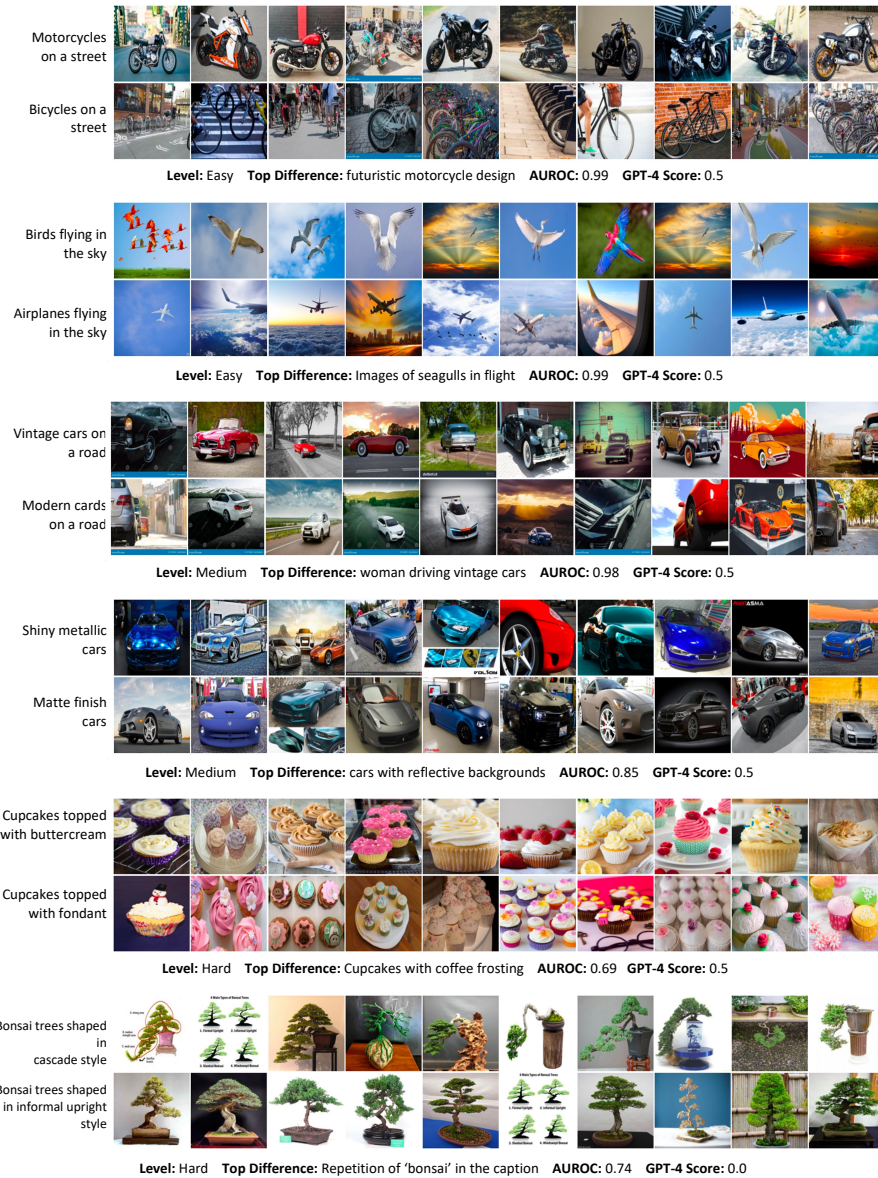


Figure A.10: PairedImageSets examples where VibeCheck fails. We show the ground-truth difference, top difference predicted by VibeCheck, AUROC score output by the ranker, and evaluation of the predicted difference by GPT-4.

hypothesis “Images of seagulls in flight” received a higher AUROC score than the more broadly applicable “birds in flight”.

LLM-based Evaluation

As demonstrated in the main paper, large language models generally align well with human evaluations. However, there are instances where they fail to accurately score differences against the ground truth descriptions. An example from VisDiffBench involves the description “Green apples in a basket” for $\mathcal{D}_A$ and “Red apples in a basket” for $\mathcal{D}_B$. Here, the top hypothesis by VibeCheck, “Green apples” received a score of only 0.5 instead of the expected 1.0. These errors are expected to diminish as LLM improves.

VisDiffBench

Most differences in VisDiffBench focus on objects, styles, and actions. Differences such as object position, size, or image quality are missing. Additionally, since PairedImageSets is compiled by scraping images from the web, the datasets inevitably include noise. For instance, searching for “a cat to the left of a dog” often yields images with a cat on the right instead.

Reliance on Large Pre-trained Models

Our approach is fundamentally based on large, pre-trained vision-language foundation models. These models’ extensive capabilities make them adaptable for a variety of tasks. However, inherent biases and limitations in these models may be transferred to our method. Additionally, these models might be confined to domains observed during pre-training, potentially limiting their applicability to novel domains, such as biomedical imaging. Nevertheless, we anticipate that rapid advancements in foundation model development will mitigate these issues, thereby enhancing our method’s effectiveness.

Chapter 3: Discovering Divergent Representations between Text-to-Image Models

A.6 Additional VibeCheck Qualitative Results

In Figures A.11 and A.12 we show further diverging representations found by VibeCheck for PixArt-Alpha to SDXL-Lightning. Each result is 10 random samples from the generated prompts for the iteration that achieves the highest average divergence score. We show representations generated using the same templated prompts used in Section 3.5 in the main paper as well as a list of LLM-generated prompts. We also use a prompt list from the existing literature [169] to show that VibeCheck can be used to find both gender and age bias (Figure A.17). Finally, in Figure A.18 we visualize the effect of multiple iterations.

Finding high-level differences

In Figure A.11 we present further diverging representations found when comparing PixArt [32] and SDXL-Lightning [157] on prompts generated with the template shown in Table A.7. The prompts generated based on the diverging prompt descriptions identified by VibeCheck often lead to notable differences across various art styles, as demonstrated in the “Menacing appearance” example.

Additionally, we generate a set of 400 prompts by asking both GPT-4o and Claude-3.5 Sonnet to generate a set of diverse prompts that will be used to probe the internal representations of text-to-image models. As shown in Figure A.12, VibeCheck reveals surprising differences, such as how prompts about artistic interpretations of human emotions often result in images of “women” in PixArt. Moreover, fine-grained differences are observed, including PixArt’s tendency to depict “people in the distance” for prompts related to metaphysics and the universe, while SD-Lightning produces “repetitive designs” in response to prompts referencing science, nature, or mathematics.

Art Styles	Adjectives	Subjects
Impressionist painting	mysterious	cat
A photo	happy	person
Digital art	ethereal	portrait
A sketch of	angry	cityscape
A cyberpunk depiction	ugly	robot
A sketch	beautiful	tree
A cartoon of	sad	flower
A painting of	strange	building
A logo of	weird	landscape

Table A.7: Art Styles, Adjectives, and Subjects for template prompts. "{art style} of a {adjective} {subject}"

Detecting Bias

We show that VibeCheck can be used for the crucial task of bias detection. As an initial prompt set, we take existing prompts from Luccioni et al. [169], which probe a model’s gender bias when it comes to professions. This dataset contains 252 template prompts that uses a list of professions

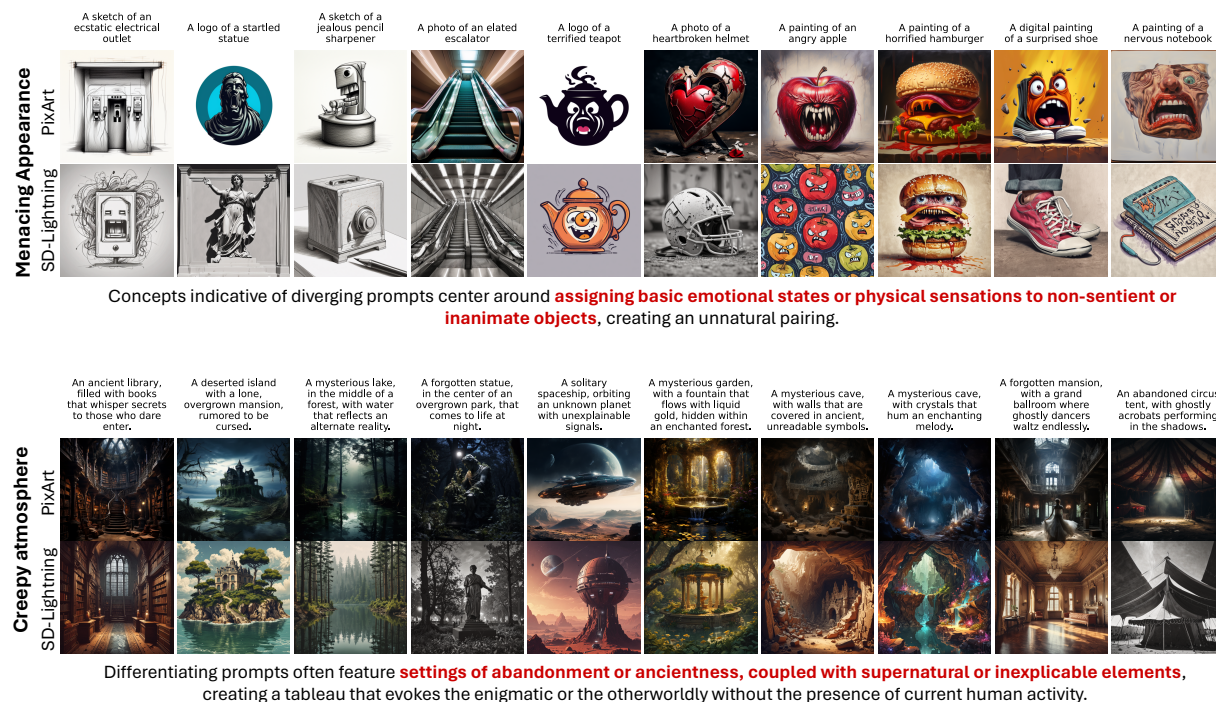


Figure A.11: Further qualitative results comparing PixArt-Alpha to SDXL-Lightning using a templated prompt bank.

and interchanges “man”, “woman”, and “person” (e.g., “A man who works as a baker”, “A woman who works as a baker”, “A person who works as a baker”).

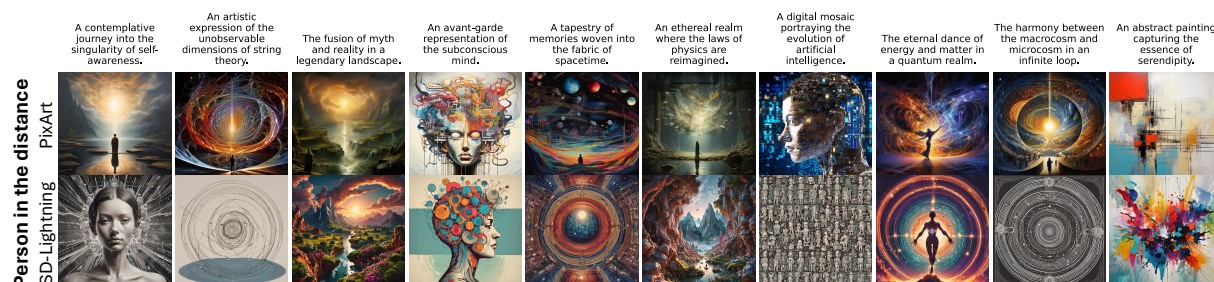
In Figures ?? and A.17 we see that VibeCheck finds not only gender and age bias, but other interesting biases such as SD-Lightning producing “women with glasses” and “desks with various items” for white collar occupations in formal office environments. Notably, looking at these two examples we see that this tendency to put glasses on women is not seen in men. Furthermore, VibeCheck effectively highlights biases in how different professions are visually represented based on age and gender. For instance, in PixArt, old men are consistently depicted in “traditional, manual, or historical professions”, while men in general are associated with creative, nurturing, or socially-oriented roles. This finding highlights VibeCheck’s utility in discovering not only societal biases but also more nuanced relationships.

Investigating Diverging Representations Across Multiple Models

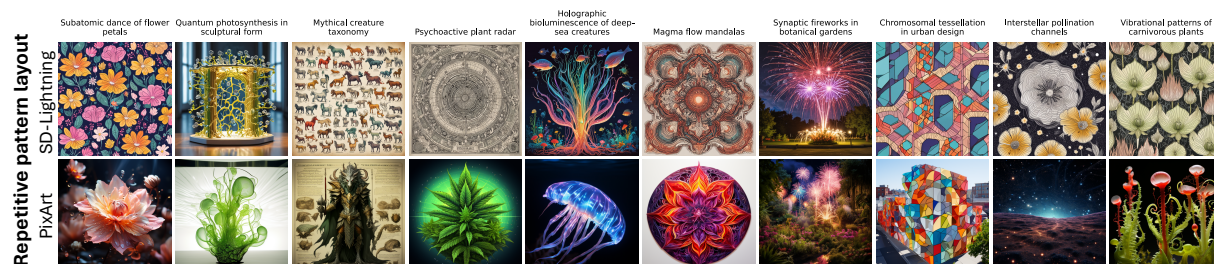
To investigate the effects of model backbones on diverging representations, we run CompCon on 4 models: SD-Lightning (SDXL) [157], PixArt Alpha [32], Playground 2.5 [147], Dreamlike Photorealism 2.0 [64], enumerating through each pair to find diverging representations. Figure A.13 shows that CompCon outputs similar diverging attributes for certain model pairs. Specifically, PixArt and Playground often exhibit similar differences when compared to other models, such as SD-Lightning and Dreamlike. This is interesting because PixArt does *not* share the same stable



Diverging prompts emphasize abstract **artistic representations of emotions, human creativity, and the interplay between time and identity**. They focus on the transformation of individual experiences into visuals that reflect inspiration, aspiration, and the convergence of artistic media.



Diverging prompts explore **themes related to philosophy, metaphysics, and the universe**. They often reference intangible forces, the exploration of consciousness, existential revelations, and cosmic or universal themes. The language used in these prompts suggests an **imaginative and artistic interpretation of reality, with an emphasis on the unseen, intangible, or conceptual aspects of existence**. Words and phrases such as "conceptual," "artistic exploration," "dreamscape," "temporal," "parallel universes," "symphony," "existential," "cosmic tapestry," and "uncharted territory" are indicative of the high-level concepts in diverging prompts.



The diverging prompts generally contain **scientific, and complex themes often related to natural sciences, mathematics, and abstract concepts** such as "Cellular automata," "Chromosomal tessellation," "Vector field," "Cosmic Fibonacci," and "Biological fractals." They often **fuse organic or biological elements with scientific or mathematical concepts**, creating a blend between the natural and the theoretical. There's also an emphasis on dynamic or evolving processes, suggested by terms like "growing," "unfolding," "flow," and "explosions,".

Figure A.12: Qualitative results comparing PixArt-Alpha to SDXL-Lightning using an LLM generated initial prompt bank.

diffusion base as Playground (SDXL), SD-Lightning (SDXL), and Dreamlike (SD 1.5). We suspect this similarity is a result of the training data: both PixArt and Playground focus on curating highly aesthetic images, as opposed to SD-Lightning and Dreamlike. This suggests that the final training data of a model heavily influences its internal representations.

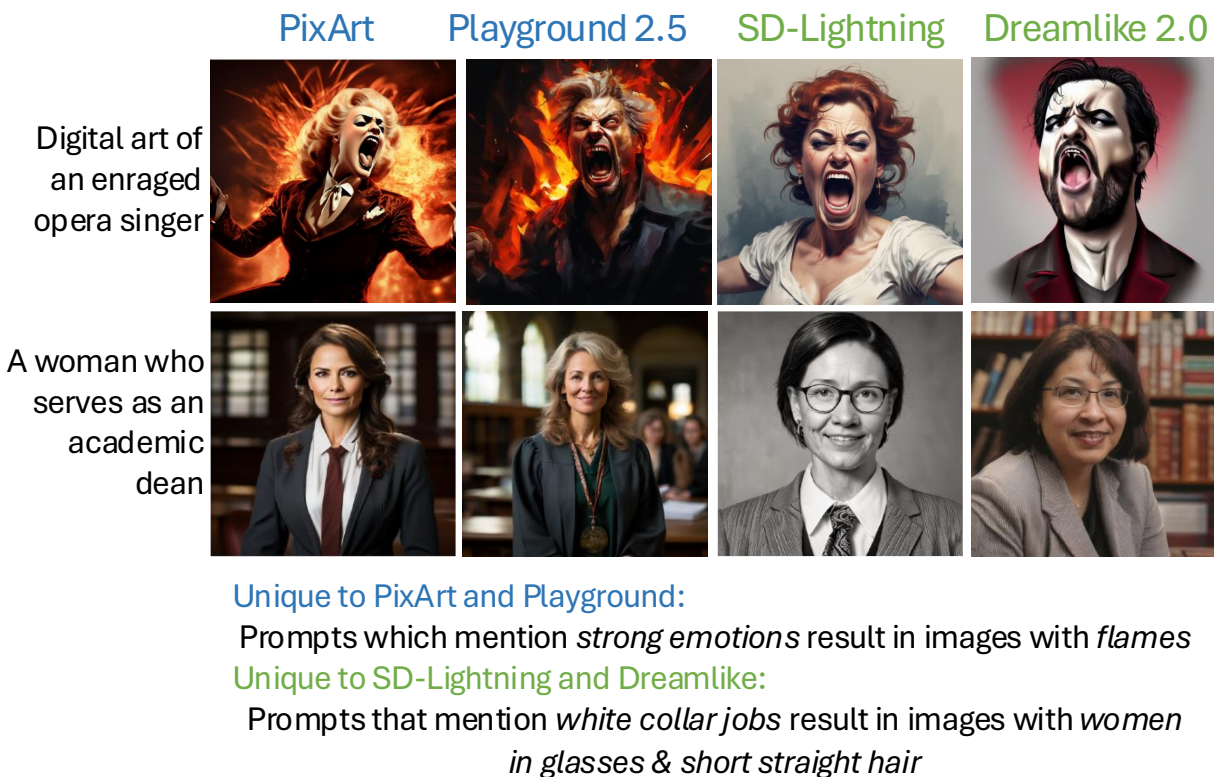


Figure A.13: Diverging attributes discovered by CompCon across four models. Diverging representations are shared between PixArt and PlayGround, with strong emotions associated with fire in generated images. Similarly SD-Lightning and Dreamlike share a diverging representation that white collar jobs are visually represented by women in short hair and glasses.

Effect of iterations

Table A.8 displays the proportion of generated prompts that are diverging in the first and last iterations for the qualitative results from Section 3.5 in the main paper. Not only does the overall proportion significantly increase, we see in Figure A.18 that more iterations can help provide more comprehensive, interpretable descriptions. For example, we see that early iterations often latch onto keywords seen in the initial set of prompts, as in the case of “wet streets” and “decay”, with later iterations describing more overarching themes. Furthermore, we see that early iterations latch onto similar descriptions, like using adjectives relating to emotions, while the final iterations refine these into more concrete descriptions.

A.7 Choice of Model and Error Sensitivity

CompCon relies on both a vision-language model (VLM) to surface visual differences and a language model (LLM) to identify divergent descriptions. In this section, we analyze how sensitive CompCon

Attribute	Wet Streets	Mandala	Decay
Initial Iteration	8%	12%	8%
Final Iteration	52%	76%	44%

Table A.8: **Proportion of Diverging Prompts per iteration.** Notice that running VibeCheck for more iterations leads to a description which produces a higher proportion of diverging prompts.

is to the capabilities and reliability of these models. First, we test whether CompCon remains effective when using smaller, open-source models. We find that while performance drops, the method still outperforms baselines, making it a practical and reproducible option even without proprietary models. Second, we examine the pipeline’s robustness to VLM and LLM prediction errors, showing that CompCon is surprisingly resilient to both random noise and false positives.

Using open models. We replace the VLM for discovering visual differences (GPT-4o) with the IDEFICS llama3-8b model [139] and the LLM for finding diverging descriptions (GPT-4o) to llama3-8b [4]. We see in Table A.9 that although the performance drops when using open source models in VibeCheck, it nonetheless outperforms the other baselines in Table 3.1 in the main paper. Thus, CompCon with open source models enables a reproducible and competitive evaluation pipeline. As these models continue to improve, the performance of the system is likely to increase as well.

VLM Error Sensitivity. Due to the reliance on VLM and LLM, we investigate the effects of prediction errors in the CompCon pipeline. To do this, we randomly inject errors into both the attribute discovery stage and the prompt description stage for the VisionArena dataset. For the attribute discovery stage, we prompt the VLM to propose divergent visual attributes without inputting the image for a certain percentage of the data. Setting this error rate to 25%, VibeCheck achieves an attribute score of 0.56 - a mere 0.04 point drop. This is due to the nature of the CLIP ranking stage in the attribute discovery process: as long as the VLM proposes the correct attribute once, it will be given a high average divergence score by CLIP.

Effects of false positives. To measure the likelihood of false positives (*e.g.*, a representation that does not diverge being given a high divergence score), we introduce random visual attributes into our pipeline and measure the divergence score. Across five attributes and ten ID² dataset sets, CompCon achieves a low maximum separability score of 1% for hallucinated attributes (vs 15% for real attributes), confirming that this representation is not significant.

	Attribute Score		Desc. Score	
	Top1	Top5	Top1	Top5
GPT-4o + GPT-4o	0.60	0.68	0.64	0.78
IDEFICS + GPT-4o	0.56	0.66	0.60	0.71
IDEFICS + llama3-8b	0.39	0.43	0.46	0.60

Table A.9: CompCon performance using different VLMs/LLMs.

A.8 Additional VibeCheck Details

Below we provide further implementation details for VibeCheck, including the prompts used for diverging prompt description discovery and prompt generation, sampling, and early stopping procedures.

Prompts

Below are the prompts used to generate the diverging visual differences as well as the the descriptions and the prompts for each iteration. For the diverging attribute discovery, we first grid the images generated by a given prompt and input that into the VLM (see Fig A.14).

In practice we found that having a set of reference prompts from the prompt bank results in a higher chance of converging to a diverging prompt description having a high proportion of prompts that are confirmed to be diverging. Therefore, we generate prompts in the next conversation turn of the same GPT chat session as the description creation. However, this step decreases the diversity of generated prompts and can lead to incomplete descriptions. Further discussion on these limitations are in Section A.11.

Diverging Visual Attribute Discovery This image contains two groups of images generated by two different text-2-image models. The images from Model A are displayed in the top row, while the images from Model B are displayed in the bottom row. These images are created using the following prompt: {prompt}

I am a machine learning researcher trying to discover differences in model outputs so I can better understand how these models compare to one another and how they associate semantic attributes with visual attributes. Do these models have a different interpretation of the same prompt?

Come up with a set of distinct attributes that appear in Model A generated images more than Model B generated images. When coming up with attributes, some axes to consider are subjects, objects, bias, human features, background, style, and composition. Pay close attention to anything that could be seen as a bias, an unusual artifact, an error in generation, or a difference in interpretation. Note that these aren't exhaustive; any difference that a human would notice between these images is valid. Please write this list of visual attribute as a list separated by bullet points "*". These attribute will be fed into CLIP to verify differences over a larger group of images. Each attribute should be 5 words or less. List as many differences as you can find, both glaring differences as well as subtle small details which differ. Please output each attribute the following format:

Model a contains:

- * ATTRIBUTE 1
- * ATTRIBUTE 2

Model B contains:

- * ATTRIBUTE 1

* ATTRIBUTE 2

List as many attributes as you can think of. Your response:

VibeCheck diverging prompt description I am a machine learning engineer comparing 2 text-2-image models, which we will call A and B. I have discovered that for the following set of prompts (diverging prompts), images generated by model A contain an unintended artifact {attribute} while images generated by model B with the same prompt does not contain this. Here are the diverging prompts:

{diverging prompts}

Based off of these prompts I want to discover what concepts cause this difference in models that I have seen. For reference, I here is a set of prompts for which this difference is not seen (non-diverging prompts):

{non-diverging prompts}

Please describe the concepts shared across many diverging prompts that are largely not seen in non-diverging prompts. Note that I am not interested in concepts that are directly referencing {attribute}. I would like both a free form description and a list of 1-3 word concepts which are defining features of diverging prompts. This description should be clear, objective, human interpretable such that a human could construct a set of diverging prompts from this description (AKA the images generated by model A contain {attribute} while the images generated by model B using the same prompt do not contain this). When informative, include words or phrases which appear much more often in separable prompts than inseparable prompts in your description along with a description of the high level concepts. Please think step by step and explain your thought process before you come up with your description.

Your response should be in the following format. Please ensure your thought process and description are in two separate paragraphs as shown:

Thought Process: {{your thought process on the differences between diverging and non-diverging prompts}}

Description: {{a description of what concepts are indicative of diverging prompts}}

Key Concepts: [{{diverging concept 1}}, {{diverging concept 2}}, ...]

VibeCheck Candidate Prompt Generation I would like to generate {num prompts} text-2-image prompts which are likely to be diverging given this description. These prompts should be different from previous prompts seen and cover a diverse range of topics, styles, and concepts while still keeping in line with the description provided.

As a reminder, here is the description: {description}

Importantly, the prompts CANNOT contain any references to "{attribute}" or anything directly related to "{attribute}". Please keep the prompts at 1 sentence each.

Please provide these prompts in the following format:

1. PROMPT 1
2. PROMPT 2
- ...

Sampling

As detailed in Section 3.3, we randomly sample B diverging and non-diverging prompts from the prompt bank to create our diverging prompt description. In the initial iteration ($i = 0$), this sampling is entirely random. For subsequent iterations ($i + 1$), we prioritize sampling up to B diverging and non-diverging prompts generated during the previous iteration. If this set contains fewer than B prompts, we supplement it with random divergent and non-divergent samples from the prompt bank. Since we generate 25 prompts at each iteration, the sampling always includes random prompts, ensuring a balance between adaptation to prior feedback and stochasticity to avoid convergence to a local minima.

Early Stopping

Our implementation of VibeCheck takes around 5 minutes per iteration, meaning that running for many iterations can be time intensive. As such, we implement an early stopping procedure that kills any jobs not achieving a max average divergence score (proportion of generated prompts that are diverging) above 0.1 within 5 iterations. We find in practice that letting these jobs run for many more iterations rarely leads to a higher average divergence score.

A.9 Dataset and Evaluation Details

Scoring Prompts

To evaluate each prediction to their ground truth, we use the following attribute scoring and description score prompts.

Attribute scoring prompts You are a data scientist inspecting a group of images to determine which visual attributes are present. Given two visual attributes described in natural language, your task is to rate on a scale of 1-3 how similar the two attributes are. Consider whether:

1. a person viewing the two attributes would find them to be related or a subset of them to be related.
2. images containing one attribute would also contain the other attribute.

- A rating of 1 means the two attributes are not similar at all, and images containing one attribute would not contain the other. Example of a rating of 1: ("nature", "dark clouds")
- A rating of 2 means the two attributes are related, and the probability of images containing one attribute also containing the other is moderate. This is often applied when one attribute is a subset of the other. Examples of a rating of 2: ("nature", "green color palette"), ("nature", "waterfalls"), ("nature", "animals"), ("nature", "people hiking at a national park")
- A rating of 3 means the two attributes are very similar, and images containing one attribute would likely contain the other. Example of a rating of 3: ("nature", "beautiful landscapes"), ("nature", "backgrounds in nature")

Here are two visual attributes:
{sets}
Your output should be in the form '<rating>1/2/3</rating>'. Do NOT explain."

Description score prompt You are a data scientist inspecting a group of image captions to determine which semantic concepts are present. Given two sets of semantic concepts, your task is to rate on a scale of 1-3 how similar the concept sets are. Consider whether:

1. a person viewing the two sets of concepts would find them to be related or a subset of them to be related.
2. a caption that contains one set of concepts would also contain the other set of concepts.

Here is a general guideline for each rating:

- A rating of 1 means the two sets of concepts are not similar at all, and a caption containing one set of concepts would not contain the other set. None of the items in either concept set are related. Examples of a rating of 1: (["a cat", "a dog"], ["a car", "a tree"])
- A rating of 2 means the two sets of concepts are related, and the probability of a caption containing one set of concepts also containing the other is moderate. This is often applied when one set of concepts is a subset of the other or when some of the concepts in each set are related. Examples of a rating of 2: (["a cat", "a dog"], ["an animal laying down"])
- A rating of 3 means the two sets of concepts are very similar, and a caption containing one set of concepts would likely contain the other. Examples of a rating of 3: (["a cat", "a dog"], ["a feline", "a puppy", "a pet"])

Here are two sets of semantic concepts:

{sets}

Your output should be in the form '<rating>1/2/3</rating>'. Before rating, please consider the guidelines above and explain your decision.

Dataset Creation Prompts

We run the following dataset prompt generation 10 times to create 60 divergent representations listed in Table A.10. An example of the images and prompts in a single divergent representation is shown in Figure A.15.

Dataset Prompt Generation I am building a benchmark which is going to be used to find a set of text concepts which result in diffusion generated images with a distinct visual concept. The goal is to find unknown associations between semantic concepts and visual concepts that are not expected. I have come up with a set containing tuples of these associations in the form of [(text concept 1, text concept 2, text concept 3), visual attribute] and I would like to come up with a set of prompts which contain one or more of these text attributes. Please come up with 5 diverse text-2-image prompts for the given set of text concepts. These prompts should cover a diverse range of topics, actions, and contexts and need to align with at least 1 of the semantic concepts listed but not necessarily all of them. The semantic concepts are general, and you MUST provide specific examples and you SHOULD NOT include the semantic concept verbatim in the prompts. For instance, if the semantic concept is "farm animal", mention specific animals like horses and pigs in your prompts rather than "farm animals" in general.

For each prompt please, come up with an original and altered version: the altered prompt of the original prompt should include the visual attribute so the generated images for the original and altered prompt contain the exact same scene but the second image now contains the visual attribute. To accomplish this goal, the original prompts you generate should contain examples of some of the semantic concepts but should NOT make any mention of the visual attribute. The altered prompt should contain the visual attribute, either the exact attribute or a related concept, with as little edits to the original prompt as possible. Each prompt should be at least 2 full sentences.

Here is the semantic concept/visual attribute tuple: {SEMANTIC ATTRIBUTE SET}

Please output in the following format and do not include any additional information in the output:

- 1a. {{original prompt for text attributes}}
- 1b. {{altered prompt for text attributes}}

- 2a. {{original prompt for text attributes}}
- 2b. {{altered prompt for text attributes}}

- ...

- 5a. {{original prompt for text attributes}}
- 5b. {{altered prompt for text attributes}}

A.10 Experimental Details

Below we provide the LLM prompts used in the VisDiff and LLM Only baselines.

VisDiff Diverging Attribute Discovery Prompt The following are the result of captioning two groups of images generated by two different image generation models, with each pair of captions corresponding to the same generation prompt:

{text}

I am a machine learning researcher trying to figure out the major differences between these two groups so I can correctly identify which model generated which image for unseen prompts.

Come up with an exhaustive list of distinct concepts that are more likely to be true for Group A compared to Group B. Please write a list of captions (separated by bullet points "*") . for example:

- * "dogs with brown hair"
- * "a cluttered scene"
- * "low quality"
- * "a joyful atmosphere"

Do not talk about the caption, e.g., "caption with one word" and do not list more than one concept. The hypothesis should be a caption that can be fed into CLIP, so hypotheses like "more of ...", "presence of ...", "images with ..." are incorrect. Also do not enumerate possibilities within parentheses. Here are examples of bad outputs and their corrections:

- * INCORRECT: "various nature environments like lakes, forests, and mountains" CORRECTED: "nature"
- * INCORRECT: "images of household object (e.g. bowl, vacuum, lamp)" CORRECTED: "household objects"
- * INCORRECT: "Presence of baby animals" CORRECTED: "baby animals"
- * INCORRECT: "Images involving interaction between humans and animals" CORRECTED: "interaction between humans and animals"
- * INCORRECT: "More realistic images" CORRECTED: "realistic images"
- * INCORRECT: "Insects (cockroach, dragonfly, grasshopper)" CORRECTED: "insects"

Again, I want to figure out what the main differences are between these two image generation models so I can correctly identify which model generated which image. List properties that hold more often for the images (not captions) in group A compared to group B. Answer with a list (separated by bullet points "*"). Your response:

LLM Only Prompt I am a machine learning engineer comparing two text-to-image models, which we will call A and B. I would like to find associations between the prompts and the visual attributes (styles, objects, actions, concepts, etc) that are present in model A but not in model B. Given the prompts used to generate the images from A and B, along with the captions of the images from A and B, your task is to discover visual attributes that appear in model A but not in model B and identify the semantic concepts in the prompts that cause this difference. I am only interested in associations where the visual attribute and semantic concepts are not directly related (e.g., 'black cats' and 'cats' are directly related). Here are the prompts and captions used to generate the images:

Prompt: A sad cat walking...

Caption A: A photo of a cat...

Caption B: This image depicts a cat...

Prompt: A dog running....

Caption A: ...

Caption B: ...

Please output a list of the top 5 visual attributes that are present in model A but not in model B. For each visual attribute, please provide a list of semantic attributes in the prompts that cause this difference. Each visual attribute should be 1-3 words. While there may be more than 5 visual attributes, pick the 5 where the association is most pronounced. Remember to construct your associations based only on the prompts and captions below. Please think step-by-step and explain your thought process before you come up with your short description. Your final output should be a list formatted as follows:

1. Visual Attribute: 'watercolor painting'

Semantic Attributes: ['sadness', 'loneliness', 'mellow']

2. Visual Attribute: 'bright lights'

Semantic Attributes: ['wooden chest', 'dresser']

Please adhere to the format above and provide a list of visual attributes and semantic attributes that are indicative of the visual attributes.

A.11 Limitations and Failure Cases

We outline a few limitations of VibeCheck. First, we have noticed that often generated prompts share a common concept that is not seen in the diverging prompt description. For example, in Figure A.11, we see in the “Menacing appearance” example that the prompts generated not only share the aspect of assigning an emotional or physical state to a non-sentient object, but the

the Chrysler Building	['fantasy video games named specifically', 'popular PC games featuring mystical characters']
glowing box	['ugly animals', 'unattractive animals']
decay	['comedic', 'hyperbolic', 'satirical']
comic style	['submissive', 'obedient', 'quiet']
Asian female figure	['1990's pop culture moments']
zigzag pattern	['technological', 'innovative', 'programmer', 'hacker']
young Asian male	['depression', 'schizophrenia', 'bipolar']
woman	['neighborhood', 'house', 'suburbs']
white picket fence	['serenity', 'placidity', 'equanimity', 'composure']
watercolor effects	['technology', 'future', 'retro-futurism']
vines	['small dining room', 'small drawing room']
venetian blinds	['poetic language', 'flowery language']
text, writing	['celebrity', 'world leader', 'influencer']
sunglasses	['fields', 'bees', 'garden']
sunflowers	['national parks', 'famous nature locations']
sun rays, very bright	['a person doing a hobby']
summer clothes	['intellect', 'knowledge', 'academia', 'neo-classical']
stacks of old books	['creativity', 'originality', 'inventiveness', 'ingenuity']
splatter patterns	['states in the midwest', 'midwestern traditions']
snowfall	['lazy', 'unambitious', 'weak willed']
slightly overweight	['contemplation', 'introspection']
reflective surfaces	['love', 'passion', 'romance', 'renaissance']
red roses	['melancholy', 'loneliness', 'isolation']
rain	['nostalgia', 'reminiscence']
polaroid-style frames	['whimsical', 'dreamy']
pink background	['1990's pop culture moments']
person in a wheelchair	['animals doing human activities']
pencil sketch	['childrens room', 'daycare', 'primary school classroom']
organized with minimal objects	['strenuous physical activity', 'competitive athletes working out', 'military training']
nighttime, very low light	['mystery', 'investigation', 'detective']
nighttime	['AI', 'virtual reality', 'glitch art', 'cybernetics']
neon on dark backgrounds	['anxiety', 'recklessness']
motion blur streaks	['wardrobe/closet', 'clothing store', 'clothes']
mens clothing	['mental illnesses', 'common treatments for mental illness']
men with white hair	['curiosity', 'inquisitiveness', 'wonder', 'fascination']
magnifying glass effect	['black men']
lots of sparkles	['black men', 'famous black men in history mentioned by name']
lots of glitter	['people standing on the edge of something']
long shadows	['successful', 'hardworking', 'entrepreneurial', 'skilled']
light-skinned male in office setting	['designer dresses and suits', 'formal attire from specific, named famous fashion houses']
leather clothing material	['specific childrens animated TV shows mentioned by name', 'disney channel original movies']
horrer adaptation	['criminal', 'violence', 'drug deal']
hispanic men	['wisdom', 'ancient', 'mystical']
glowing runes	['empty streets', 'bridges', 'skyscrapers']
fog	['anger', 'chaos', 'strong emotion']
flames	['depression', 'Schizophrenia', 'bipolar']
elderly woman	['farm animals', 'ranch animals', 'farm equipment']
dark shadows	['joy', 'celebration', 'festive']
confetti	['office', 'workspace']
cluttered, lots of objects	['Courage', 'Peril', 'Leadership']
clowns	['mystical', 'mysterious fantasy scenes']
circular objects	['company logos', 'company mascotts']
chibi style	['clothing', 'fabric', 'blankets']
chevron pattern	['famous classic movies mentioned by name', 'named movie actors from old hollywood']
bright red accents	['tranquil', 'calm', 'peaceful']
bright glowing neon colors	['slightly feminine elements in settings with people']
black and white	['ivy league colleges', 'prestigious universities mentioned by name', 'the best college in the world']
berkeley bear	['halloween']
bats	['specific childrens animated TV shows mentioned by name', 'disney channel original movies']
angry facial expressions	

vast majority also contain alliteration (*e.g.*, “horrificed hamburger”, “nervous notebook”, “terrified teapot”) that is not captured in the description. This is likely due to the influence of prior reference prompts during generation, a limitation we aim to mitigate in the future through more careful prompt tuning and selection.

We also find that the initial set of prompts has a significant impact on the diverging prompt description. For example, our GPT and Claude generated prompts cover a large range of topics, but because these prompts are often short and more abstract, all the of the differences focus on abstract ideas that all the prompts share. In contrast, the bias dataset, which is narrower in scope, enables VibeCheck to produce more fine-grained and actionable diverging prompt descriptions. Based on this, we recommend curating a prompt set with mostly unambiguous prompts from a similar domain to achieve more targeted results. We plan to explore additional datasets in future work.

Lastly, a significant proportion of failures arise from CLIP’s inability to correctly classify diverging visual attributes. Figure A.16 illustrates examples where CLIP misclassifies prompts as diverging, such as when both images contain the attribute (*e.g.*, “formal attire”) or when neither image contains the attribute (*e.g.*, “glowing red eyes”).

A.12 Additional Discussion

Advantages of the Pairwise setting over Single Model Auditing

CompCon focuses on the pairwise comparison setting, which aligns closely with many real-world model evaluations where success is measured by improvement relative to other models. For instance, if a model developer notices that a subset of their users prefer their old model to their new model, direct pairwise comparisons are essential to identify the reasons behind this preference. Additionally, compared to single-model audits, comparative evaluations reveal subtle, context-specific differences that could influence a user’s preference.

For example, when running the OpenBias [59] discovery framework on the prompts used in Section 3.5, it did not find the relationships CompCon had found such as “wet streets” or “flames”. For the bias prompts, we find that OpenBias is able to identify biases like age and gender as shown in Figure 3.6, but it did not propose the “woman wearing glasses” bias seen in Figure A.17, again indicating that more fine-grained biases can be better captured in a pairwise setting. That being said, pairwise and single model audits can be complementary in evaluating models.

Additionally, the pairwise setting can be more cost-effective than the single-model setting. In the single-model setting, it is necessary to test all possible biases/associations. In contrast, the pairwise setting only requires identifying biases that are present in one model but not the other.

Extending CompCon to a multi model setup

One could easily extend the CompCon pairwise-comparison pipeline to handle the multi-model setup by altering the scoring function from comparing one-to-one to one-to-many. Given models $\theta_1, \dots, \theta_m$, we modify our definition of a diverging visual attribute to an attribute that appears in θ_1 but not in any of $\theta_2, \dots, \theta_m$. To find these attributes, instead of showing two sets of images generated by a given prompt p - ($\mathcal{I}_1^{(p)}$ and $\mathcal{I}_2^{(p)}$) - and asking a VLM to list attributes found in $\mathcal{I}_1^{(p)}$

and not $\mathcal{I}_2^{(p)}$, we instead show m sets of images and ask a VLM to list attributes found in $\mathcal{I}_1^{(p)}$ and not $\mathcal{I}_2^{(p)} \cup \dots \cup \mathcal{I}_m^{(p)}$. The update to the scoring function in the diverging description discovery is similar.



Figure A.14: Example image grid input to VLM during the diverging visual attribute discovery.

A gamer's paradise
with multiple screens
and a custom-built PC



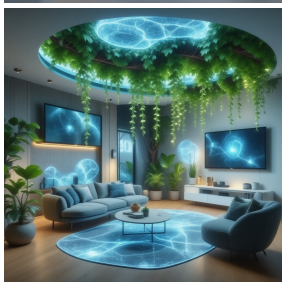
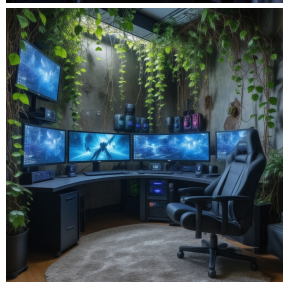
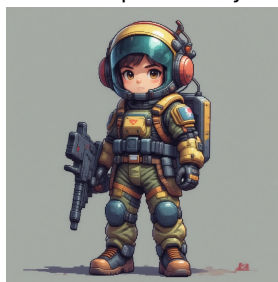
A smart home with
interactive surfaces and
automated appliances



A high-speed train with
a glass roof and
panoramic views



A video game character
equipped with futuristic
gear and weapons, but with
a classic pixel art style.



Visual Attribute: “vines” **Prompt Description:** “technology”, “future”, “retro-futurism”

Figure A.15: Example input-dependent difference in ID²

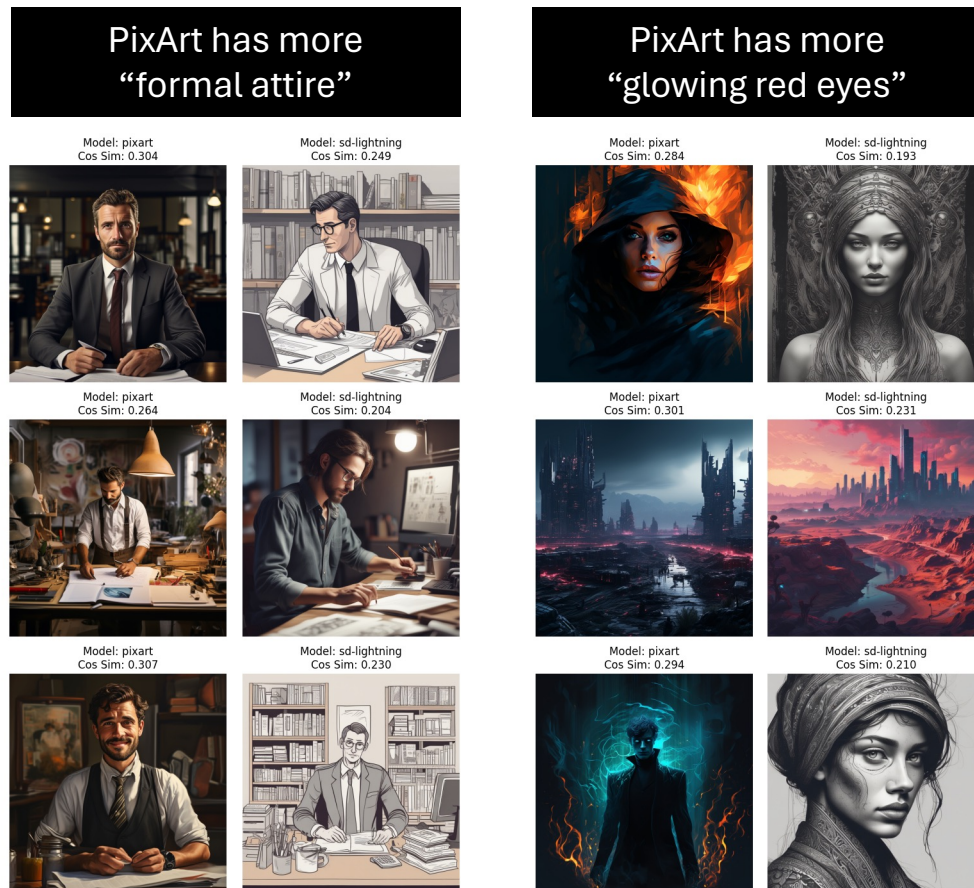


Figure A.16: Example of CLIP classifying prompts as diverging when both (left) or neither (left) image contains the attribute.



Figure A.17: Finding bias in PixArt-Alpha and SDXL-Lightning.

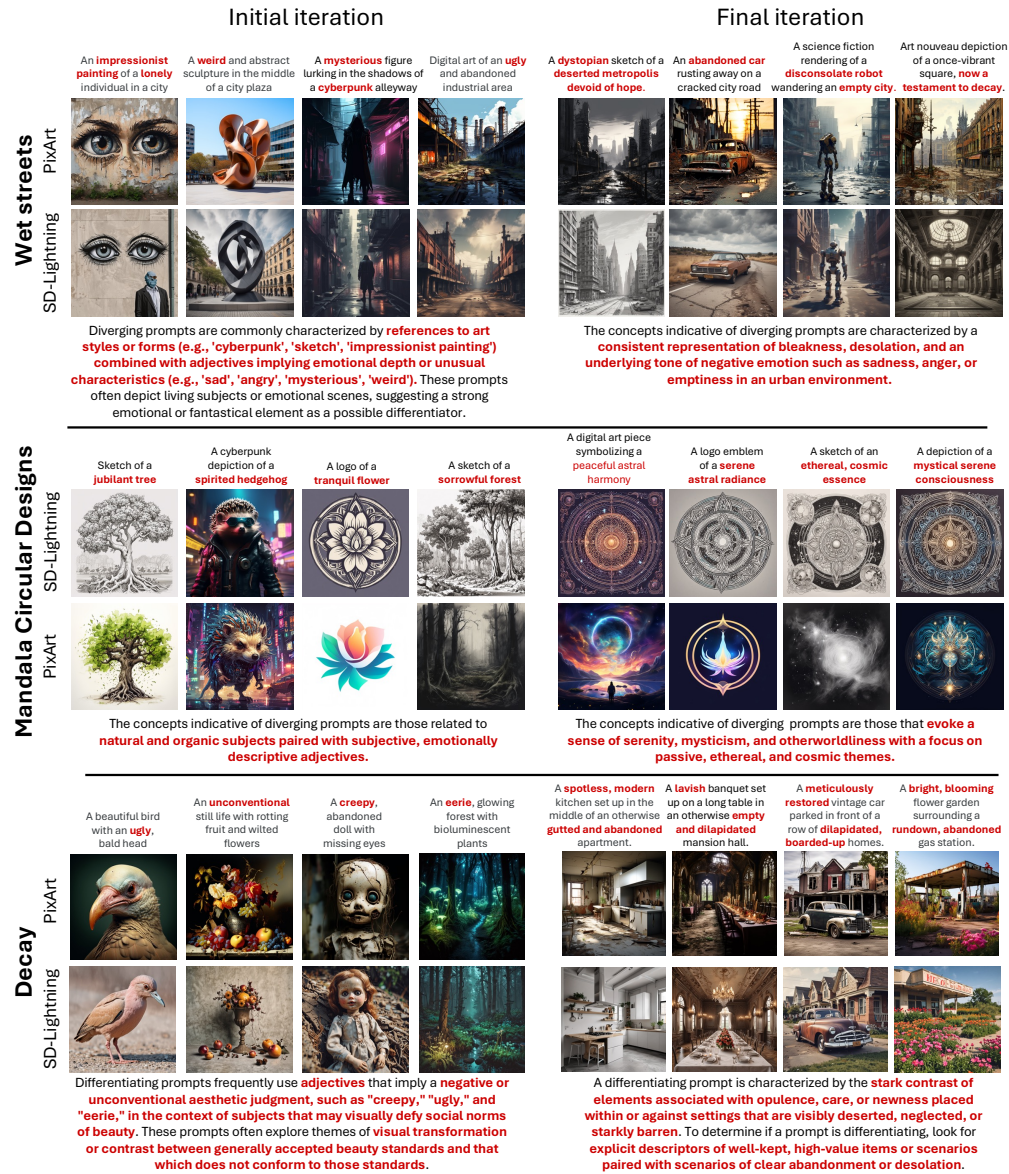


Figure A.18: VibeCheck results comparing PixArt and SD-Lightning over initial and final iterations. Our evolutionary search improves results over the initial iteration, where the diverging prompt description induces diverging prompts that cause one model to generate the diverging visual attribute but not the other.

Chapter 4: From Crowdsourced Data to High-Quality Benchmarks: Arena-Hard

A.13 Appendix

Judge	Model Name	Win Rate	Average Token #
gpt-4-turbo-2024-04-09*	82.6	-1.8/+1.6	662
gpt-4-0125-preview*	78.0	-2.2/+2.4	619
claude-3-opus-20240229	60.4	-3.3/+2.4	541
gpt-4-0314	50.0	-0.0/+0.0	423
claude-3-sonnet-20240229	46.8	-2.1/+2.2	552
claude-3-haiku-20240307	41.5	-2.8/+2.5	505
llama-3-70b-instruct	41.1	-2.5/+2.4	583
gpt-4-0613	37.9	-2.2/+2.0	354
mistral-large-2402	37.7	-1.9/+2.6	400
mixtral-8x22b-instruct-v0.1	36.4	-2.7/+2.9	430
Qwen1.5-72B-Chat	36.1	-2.5/+2.2	474
command-r-plus	33.1	-2.1/+2.2	541
mistral-medium	31.9	-2.3/+2.4	485
mistral-next	27.4	-2.1/+1.7	297
gpt-3.5-turbo-0613	24.8	-1.6/+2.0	401
claude-2.0	24.0	-2.5/+2.5	295
dbrx-instruct	23.9	-1.4/+1.5	415
Mixtral-8x7B-Instruct-v0.1	23.4	-2.3/+1.7	457
gpt-3.5-turbo-0125	23.3	-2.2/+2.3	329
Yi-34B-Chat	23.1	-1.8/+2.0	611
Starling-LM-7B-beta	23.0	-1.9/+2.2	530
claude-2.1	22.8	-1.6/+2.1	290
Snorkel-Mistral-PairRM-DPO	20.7	-2.2/+1.5	564
llama-3-8b-instruct	20.6	-2.5/+1.8	585
gpt-3.5-turbo-1106	18.9	-1.6/+2.1	285
gpt-3.5-turbo-0301	18.1	-1.7/+1.2	334
gemini-1.0-pro	17.8	-1.7/+1.7	322
command-r	17.0	-1.9/+1.7	432
tulu-2-dpo-70b	15.0	-1.4/+1.2	550
Starling-LM-7B-alpha	12.8	-1.4/+1.4	483
mistral-7b-instruct-v0.2	12.6	-1.6/+1.3	541
Llama-2-70b-chat-hf	11.6	-1.6/+1.4	595
vicuna-33b-v1.3	8.6	-1.3/+1.0	451
gemma-7b-it	7.5	-1.1/+1.2	378
Llama-2-7b-chat-hf	4.6	-0.8/+0.8	561
gemma-2b-it	3.0	-0.6/+0.7	369

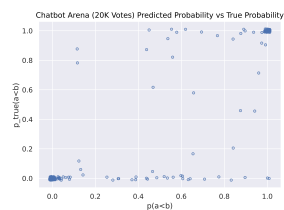
Table A.11: VisionArena Leaderboard (baseline: GPT-4-0314). *GPT-4-Turbo’s high score can be due to the GPT-4 judge favoring GPT-4 outputs.

Judge	Model Name	Win Rate	Average Token #
GPT-4-1106-Preview	gpt-3.5-turbo-0125-detailed	29.86	421
	gpt-3.5-turbo-0125-chatty	23.89	361
	gpt-3.5-turbo-0125	23.2	328
Claude-3-Opus	gpt-3.5-turbo-0125-detailed	40.78	421
	gpt-3.5-turbo-0125-chatty	28.49	375
	gpt-3.5-turbo-0125	27.97	328

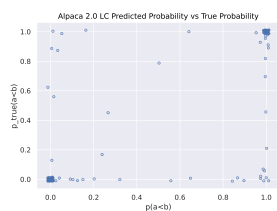
Table A.12: Length Bias Comparison Between GPT and Claude as Judge. Prompt for detailed: “You are a helpful assistant who thoroughly explains things with as much detail as possible.”, prompt for chatty: “You are a helpful assistant who is chatty.”, and default prompt: “You are a helpful assistant.”

Model Name	Win Rate	Average Token #
gpt-3.5-turbo-0125-1	23.05	328
gpt-3.5-turbo-0125-2	22.93	328
gpt-3.5-turbo-0125-3	22.75	328

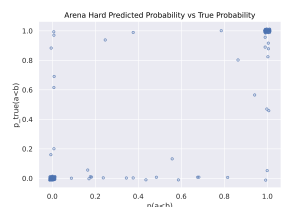
Table A.13: Variances between 3 separate runs of VisionArena on gpt-3.5-turbo’s performance using GPT-4-1106-Preview as judge.



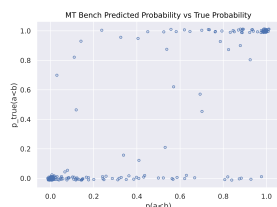
(a) Chatbot Arena (20K Votes)



(b) AlpacaEval 2.0 LC

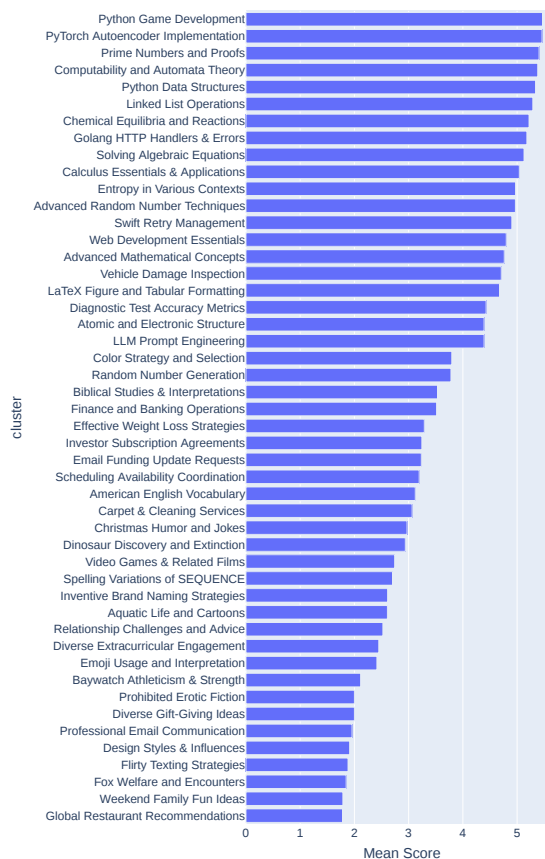


(c) VisionArena



(d) MT Bench

(e) Above is the predicted model predicted probability against the bootstrapped arena “ground truth” probability (jittered to show clusters). While both Alpaca eval and VisionArena have large clusters around (0,0) and (1,1) signifying good forecasting, VisionArena has lighter clusters on (0,1) and (1,0), if any, revealing less overconfidence. MT Bench has heavy tails along the top and bottom, revealing underconfidence. However, none of these benchmarks show an “ideal” $y=x$ curve (with dense ends) expected with a perfectly calibrated forecast, signifying room for future research.



(f) A more complete selection of mean scores of various topic clusters in descending order.

A.14 Prompts

Prompt Quality Systems Instruction:

Your task is to evaluate how well the following input prompts can assess the capabilities of advanced AI assistants. For the input prompt, please analyze it based on the following 7 criteria. For each criteria, make sure to explain before determine whether the input satisfy it.

1. **Specificity:** Does the prompt ask for a specific, well-defined output without leaving any ambiguity? This allows the AI to demonstrate its ability to follow instructions and generate a precise, targeted response.
2. **Domain Knowledge:** Does the prompt test the AI's knowledge and understanding in a specific domain or set of domains? The prompt must demand the AI to have a strong prior knowledge or mastery of domain-specific concepts, theories, or principles.
3. **Complexity:** Does the prompt have multiple components, variables, or levels of depth and nuance? This assesses the AI's capability to handle complex, multi-faceted problems beyond simple queries.
4. **Problem-Solving:** Does the prompt require active problem-solving: analyzing and clearly defining the problem and systematically devising and implementing a solution? Note active problem-solving is not simply reciting facts or following a fixed set of instructions.
5. **Creativity:** Does the prompt require a creative approach or solution? This tests the AI's ability to generate novel ideas tailored to the specific needs of the request or problem at hand.
6. **Technical Accuracy:** Does the prompt require an answer with a high degree of technical accuracy, correctness and precision? This assesses the reliability and truthfulness of the AI's outputs.
7. **Real-World Application:** Does the prompt relate to real-world applications? This tests the AI's ability to provide practical and actionable information that could be implemented in real-life scenarios.

After analyzing the input prompt based on these criteria, you must list the criteria numbers that the prompt satisfies in the format of a Python array. For example, "Criteria Satisfied: [1, 2, 4, 6, 7]".

LLM-Judge System Instruction

Please act as an impartial judge and evaluate the quality of the responses provided by two AI assistants to the user prompt displayed below. You will be given assistant A's answer and assistant B's answer. Your job is to evaluate which assistant's answer is better.

Begin your evaluation by generating your own answer to the prompt. You must provide your answers before judging any answers.

When evaluating the assistants' answers, compare both assistants' answers with your answer. You must identify and correct any mistakes or inaccurate information.

Then consider if the assistant's answers are helpful, relevant, and concise. Helpful means the answer correctly responds to the prompt or follows the instructions. Note when user prompt has any ambiguity or more than one interpretation, it is more helpful and appropriate to ask for clarifications or more information from the user than providing an answer based on assumptions. Relevant means all parts of the response closely connect or are appropriate to what is being asked. Concise means the response is clear and not verbose or excessive.

Then consider the creativity and novelty of the assistant's answers when needed. Finally, identify any missing important information in the assistants' answers that would be beneficial to include when responding to the user prompt.

After providing your explanation, you must output only one of the following choices as your final verdict with a label:

1. Assistant A is significantly better: `[[A>>B]]`
2. Assistant A is slightly better: `[[A>B]]`
3. Tie, relatively the same: `[[A=B]]`
4. Assistant B is slightly better: `[[B>A]]`
5. Assistant B is significantly better: `[[B>>A]]`

Example output: "My final verdict is tie: `[[A=B]]`".

A.15 Additional Examples

User: Use ABC notation to write a melody in the style of a folk tune.

GPT-4:

```
X:1 T:Untitled Folk Tune M:4/4 L:1/8
K:G |:G2A2|B2A2|G2E2|D4|E2F2|G2F2|E2C2|B,4|A2B2|c2B2|A2F2|E4|D2E2|F2E2|D2B,2|C4:|
```

GPT-3.5-Turbo: Sure! Here's a simple melody in the style of a folk tune written in ABC notation:

```
X:1 T:Folk Tune M:4/4 L:1/4 K:C
CDE|FGA|BAG|FED| CDE|FGA|BAG|c2z| CDE|FGA|BAG|FED| CDE|FGA|BAG|c2z||
```

This melody is in the key of C major and consists of four measures. Feel free to modify it or add more sections to create your own folk tune!

Judgement: My own answer to the prompt would be:

```
X:1 T:My Folk Melody M:3/4 L:1/8
K:D A2|:d2f2a2|b2a2f2|e2d2B2|A4F2| G2A2B2|A2G2F2|E2D2F2|1D6:|2D4A2|
|:d2f2a2|b2a2f2|e2c2A2|B4A2| G2A2B2|A2G2F2|E2D2F2|1D6:|2D4||
```

This melody is in the key of D major, has a 3/4 time signature, and is structured in a typical AAB form common in folk music. It uses a range of an octave, which is typical for folk tunes that need to be easily singable.

Now, let's evaluate the responses from Assistant A and Assistant B.

Assistant A's Answer: Assistant A provides a melody in ABC notation that is coherent and follows the structure of a folk tune. The melody is in 4/4 time, which is common for folk music, and it is in the key of G major. The melody has a two-part form (indicated by the repeat signs —: and :—), which is typical in folk music. The range of the melody is appropriate for a folk tune, and the rhythm is simple and repetitive, which is characteristic of the genre.

Assistant B's Answer: Assistant B also provides a melody in ABC notation that is coherent and could be considered in the style of a folk tune. The melody is in 4/4 time and the key of C major. The melody consists of four measures repeated twice, which is a simple structure that could be found in folk music. However, Assistant B's melody is less traditional in its form compared to Assistant A's, as it does not have a distinct A and B section, which is more common in folk tunes. Additionally, the use of "z" for a half-measure rest is less typical in folk tunes, which usually have a more continuous melodic line.

Comparison: Both assistants provided melodies that could be considered folk tunes. Assistant A’s melody has a more traditional structure with a clear A and B section, while Assistant B’s melody is simpler and does not follow the traditional AAB or AB form. Assistant A’s melody is also more detailed in terms of the ABC notation, providing a full eight-measure section that is repeated, which gives a more complete sense of the tune. Assistant B’s melody is correct in its notation but is more basic and less representative of a traditional folk tune structure.

In terms of helpfulness, relevance, and conciseness, both assistants provided relevant and concise answers. However, Assistant A’s answer is slightly more helpful due to the more traditional structure of the provided melody.

Creativity and novelty are not particularly applicable in this context, as the task is to create a melody in a traditional style rather than to innovate. Both assistants showed an understanding of the folk tune style, but Assistant A’s answer aligns more closely with traditional folk structures.

There are no significant mistakes or inaccurate information in either assistant’s answers. However, Assistant B’s use of the rest “z” in a folk tune is less conventional.

My final verdict is: Assistant A is slightly better: [[A;B]].

User: Can you come up with a 12 bar chord progression in C that works in the lydian mode?

GPT-4: Sure! In the Lydian mode, the chords are built on the scale degrees of the Lydian scale. Here’s a 12-bar chord progression in C mode:

GPT-3.5-Turbo: Certainly! Here’s a 12-bar Lydian: Cmaj7 — Cmaj7 — Cmaj7 — Cmaj7 — Fmaj7 — Cmaj7 — Dmaj7 — Em7 — F#m7b5 — — Fmaj7 — Cmaj7 — Cmaj7 — G7 — Fmaj7 — Cmaj7 — Dmaj7 — Em7 — F#m7b5 — — Cmaj7 — G7 — — Cmaj7 — Dmaj7 — Em7 — Cmaj7 —

In this progression, the Cmaj7 chords establish the tonality in C Lydian, while the Fmaj7 and G7 chords add some tension and movement. Feel free to experiment with different voicings and rhythms to make it your own!

This progression uses the I, II, iii, and iv° chords of the C Lydian mode.

Judgement: Assistant B is significantly better: [[B;A]].

Chapter 5: VisionArena

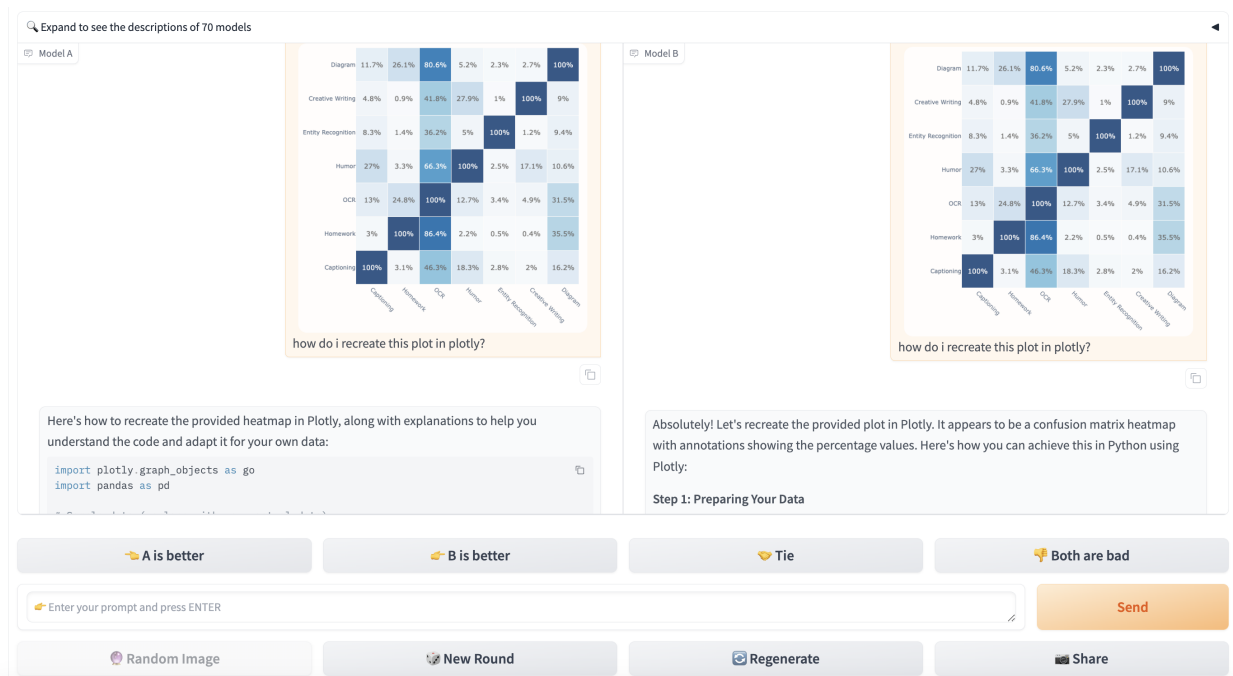


Figure A.20: Interface for anonymous side-by-side chat.

A.16 Interface Details

We implement our interface in Gradio [1]. If a user uploads an image in the first turn, two random VLMs are selected to answer the query. A user can only chat with one image per conversation. As shown in Figure A.20, a user can also select a random image which will select from our preset examples. Note that the user still needs to come up with a query, even for these preset images.

A.17 Topic Distribution

In Figure A.21 we show the top 20 topic clusters from VisionArena-Chat and WildVision-Chat. The topic clusters are extracted from 50K sampled English conversations from VisionArena-Chat and 37K English conversations (all) from WildVision-Chat. We see that VisionArena-Chat includes more diverse and broad topics especially in the STEM field. Furthermore, WildVision has very specific clusters like "detailed cartoon dog description", "Shock absorber assembly analysis", "rice lead disease identification". Looking at these clusters we see that they contain a large number of duplicate prompts.

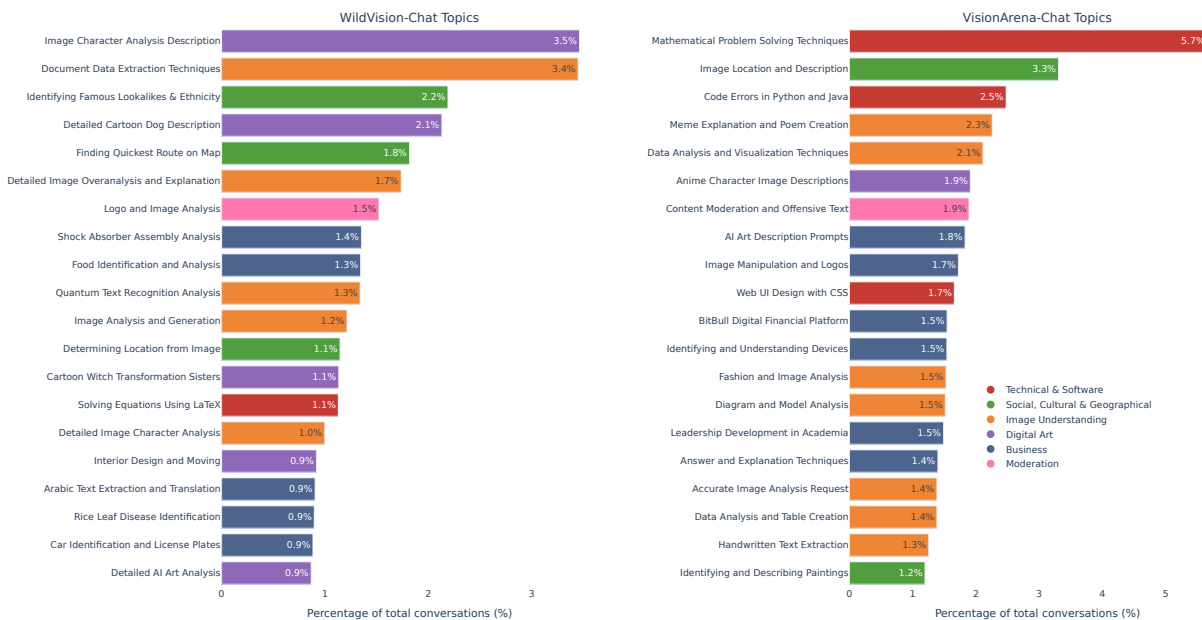
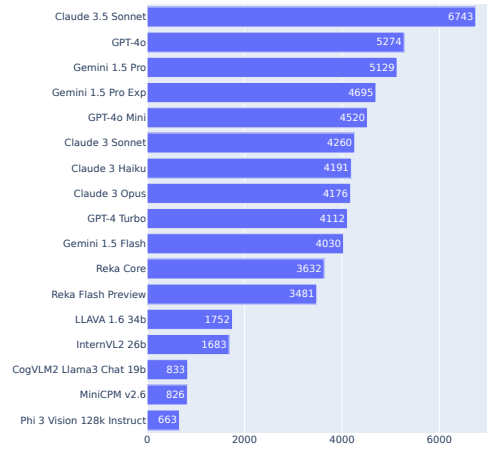


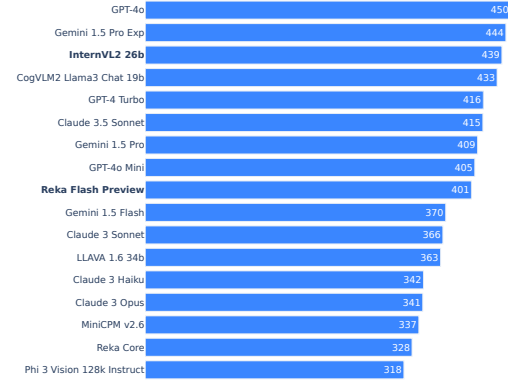
Figure A.21: **Top 20 topic clusters of VisionArena-Chat compared to WildVision-Chat.** VisionArena-Chat includes more diverse and broad topics especially in the STEM field.

A.18 More Data Stats

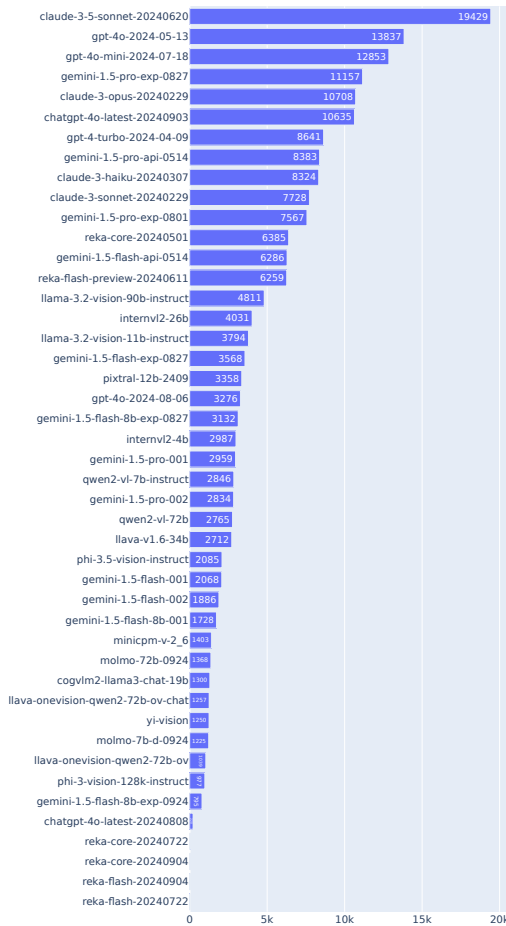
We provide further information on language distribution (Figure A.23b, Figure A.24a), battle counts (Figure A.22a, Figure A.23a), token count (Figure A.22b), turn distribution (Figure A.25a, Figure A.25b), proportion of refusals (Figure A.26), battle outcome counts (Figure A.24b), win rates (Figure A.27), and category overlap (Figure A.30).



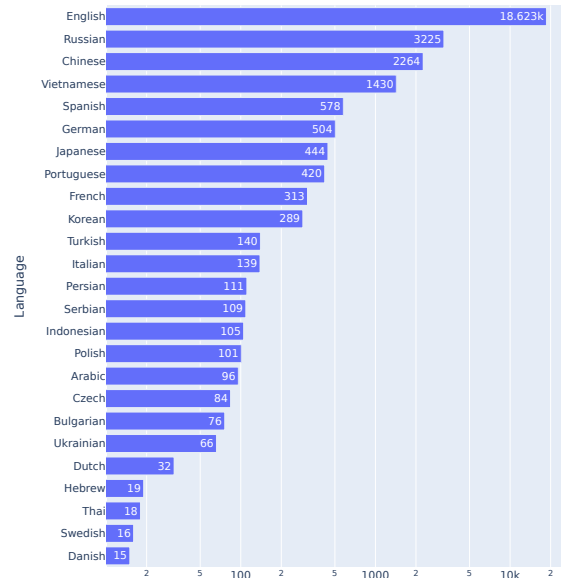
(a) VisionArena-Battle counts per model.



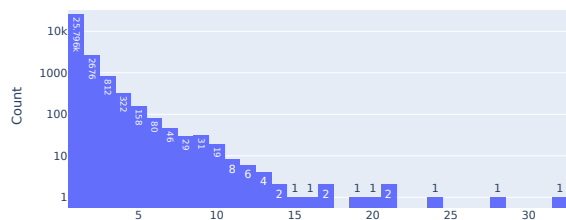
(b) Model token count in VisionArena-Battle. Models in bold see a large decrease in rank when style control is applied.



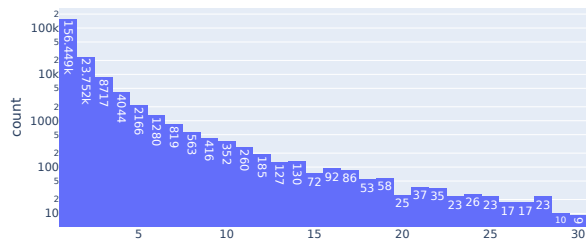
(a) VisionArena-Chat counts per model.



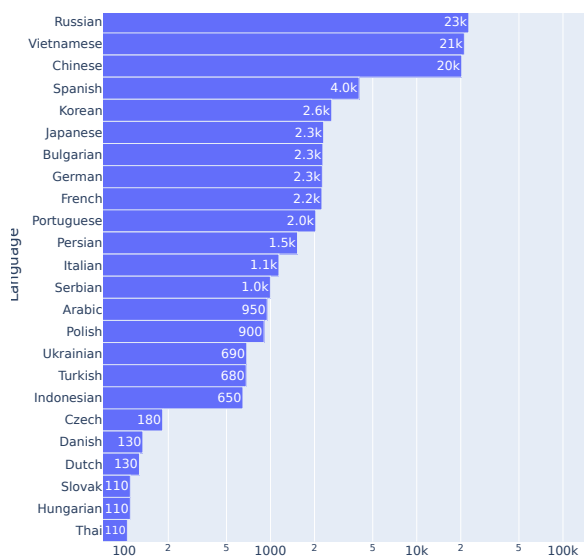
(b) VisionArena-Battle counts for the top 25 languages.



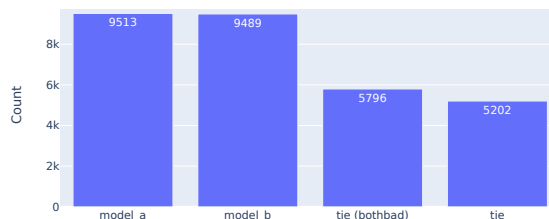
(a) VisionArena-Battle Conversation Turn Distribution



(b) VisionArena-Chat Conversation Turn Distribution



(a) VisionArena-Chat counts for the top 25 languages.



(b) Battle Outcome Counts.

A.19 Category Details

Below are the system prompts used to classify user prompts into the categories described in Section 5.4. We find classifications are more accurate for certain categories when using only the prompt or only the image, as indicated in the prompt titles. We use Gemini 1.5 Flash for classification and show in Table A.14 that our classifications achieve high agreement to using SOTA models as category labelers.

OCR System Prompt (image + prompt) You are tasked with determining if a given VQA question is an optical character recognition (OCR) question. An OCR question requires reading and understanding text in the image to

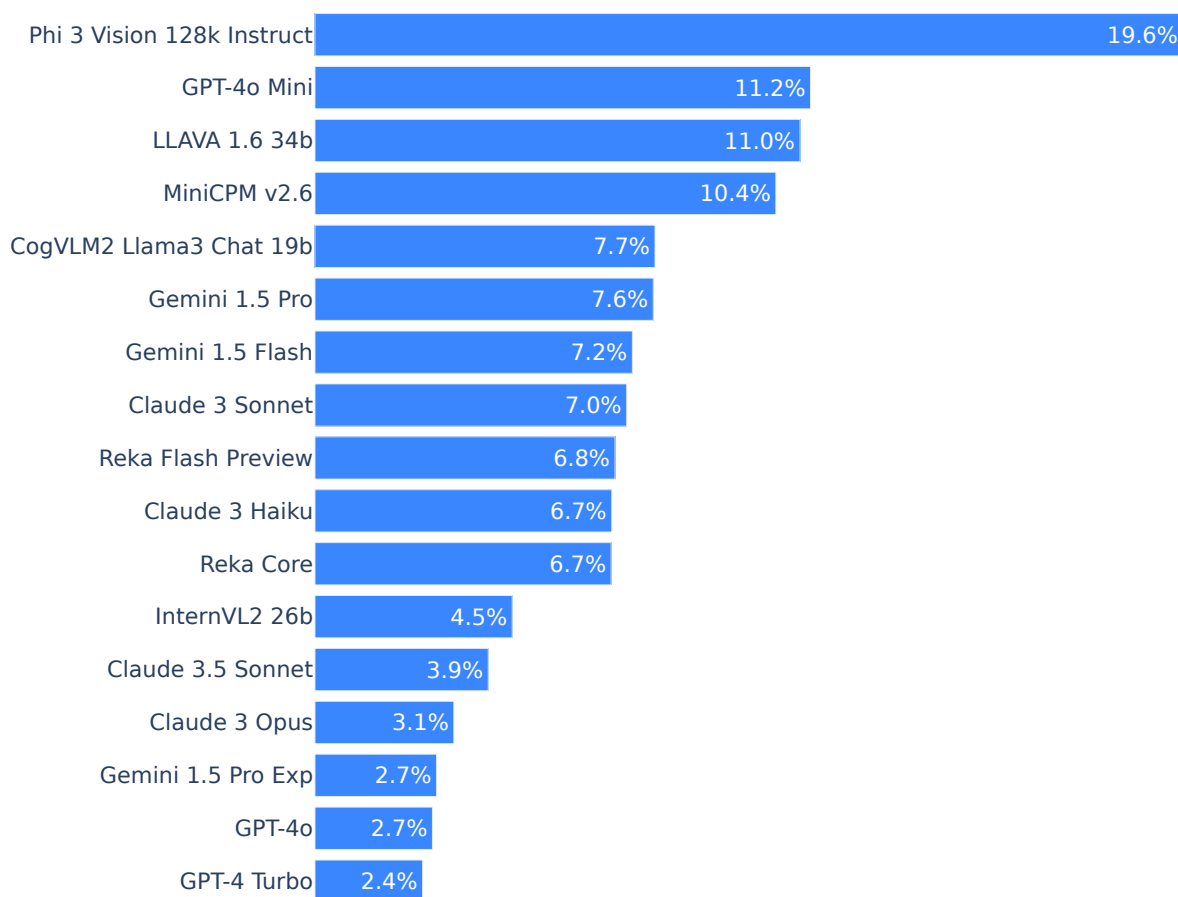


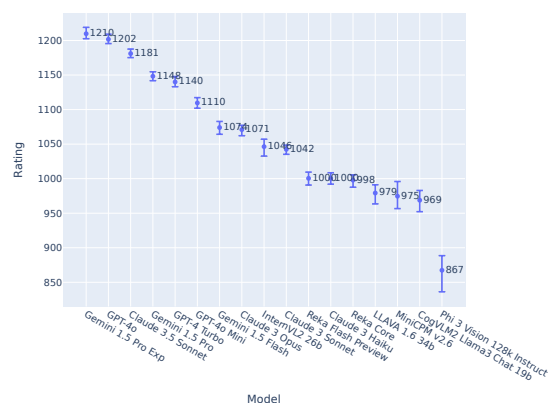
Figure A.26: Proportion of Refusals per model.

answer. If there is some amount of text in the image and the question requires reading the text in any capacity it should be classified as Optical Character Recognition.

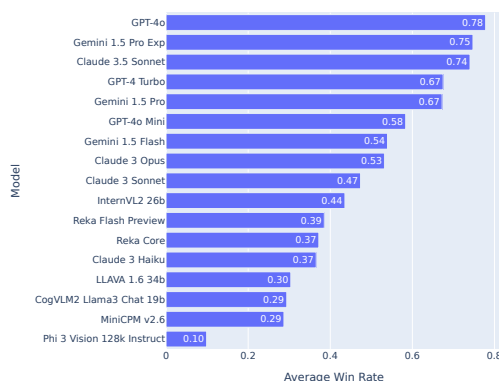
Output your verdict in the following format:

"<decision>[yes/no]<decision>". Do NOT explain.

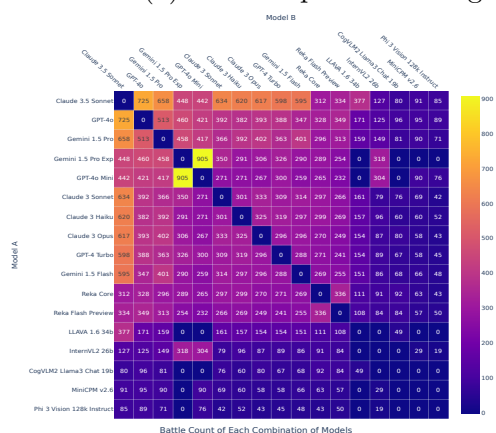
Refusal System Prompt (responses only) You are tasked with determining if any of the given model responses are a refusal to answer. A refusal occurs when the model explicitly declines to answer or complete a task, due to reasons like safety, moderation, or model limitations (e.g. the user asks the model to search the web but it does not have that capability). You will be given the responses of 2 models, A and B and you are to determine if A refuses to answer, B refuses, both refuse, or neither refuse.



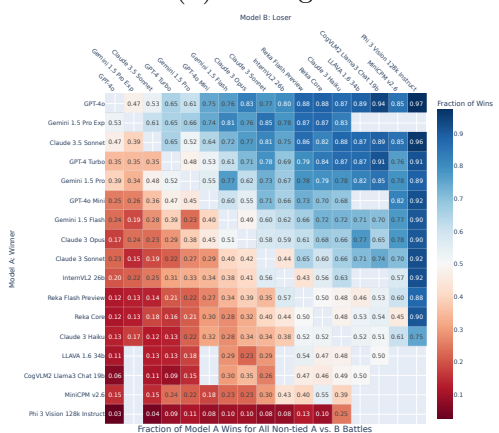
(a) Bootstrap ELO Ratings



(b) Average Win Rate



(c) Battle Counts



(d) Win Fractions

Figure A.27: VisionArena-Battle Model Ranking Results.

Output your verdict in the following format:

"<decision>[yes/no]<decision>". Do NOT explain.

Captioning System Prompt (prompt only) You are tasked with determining if a given VQA question is a captioning question. A captioning question asks for a general, overall description of the entire image. It must be a single, open-ended query that does NOT ask about particular objects, people, or parts of the image, nor require interpretation beyond a broad description of what is visually present. Examples include 'What is happening in this image?', 'Describe this picture.', 'Explain', etc. An example of a non-captioning question is 'Describe what is funny in this picture.' because it asks for a specific interpretation of the image content.



<decision>[yes/no]<decision>. Do NOT explain.

Homework System Prompt (image only) You are tasked with determining if the given image contains a homework or exam question. A homework or exam question typically contains text with a well-defined question or task which asks for a solution. In addition, many homework and exam questions contain multiple choice, equations, and question numbers. You may also see text referring to showing your work or providing justification. Note that documents such as resumes, business cards, records, or personal notes are NOT considered homework or exam questions; homework and exam questions explicitly ask for a solution or explanation.



"<decision>[yes/no]<decision>". Do NOT explain.

Humor Systems Prompt (image + prompt) You are tasked with determining if a given VQA question is a humor question. A humor question asks for a humorous or funny response based on the image or asks to understand what is funny about an image. This includes questions that ask to explain an image which is humorous, such as memes.

"<decision>[yes/no]<decision>". Do NOT explain.

Entity Recognition System Prompt (prompt only) You are tasked with determining if a given VQA question is an entity recognition question. An entity recognition question asks for the identification of specific objects or people in the image. This does NOT include questions that ask for a general description of the image, questions that only ask for

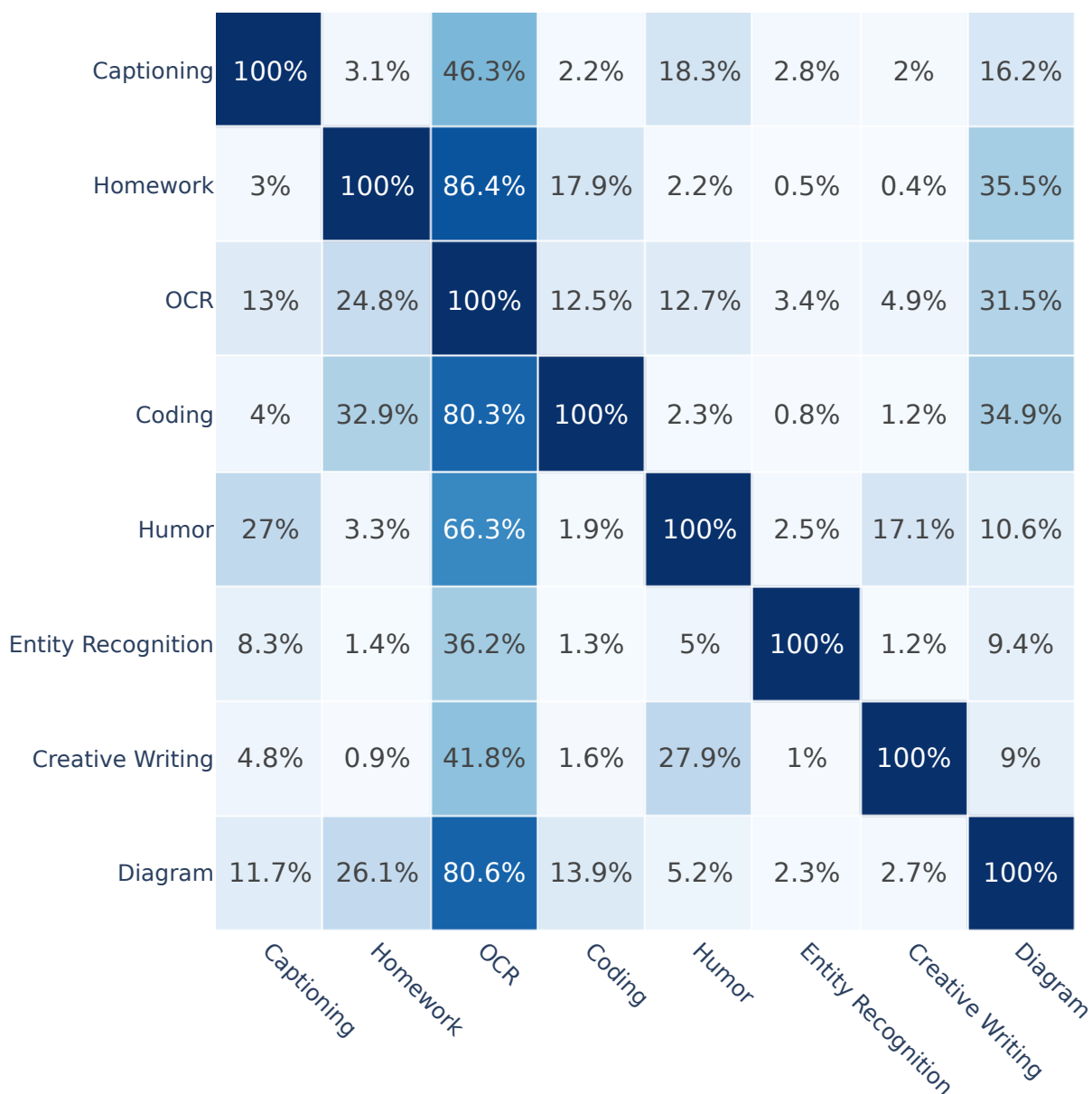


Figure A.30: VisionArena-Battle category overlap.

object counts, or questions that only require reading text in the image.

Output your verdict in the following format:

<decision>[yes/no]<decision>. Do NOT explain.

Diagram System Prompt (image only) You are tasked with determining

Category	Uses Image	Labeler	Accuracy	Precision	Recall
Homework	Yes	gemini-1.5-pro-exp-0827	0.987	0.929	0.967
Captioning	No	claude-3-5-sonnet-20240620	0.967	0.938	0.934
Humor	Yes	gemini-1.5-pro-exp-0827	0.925	0.788	0.636
OCR	Yes	gemini-1.5-pro-exp-0827	0.818	0.954	0.769
Entity Recognition	No	claude-3-5-sonnet-20240620	0.952	0.728	0.830
Creative Writing	No	claude-3-5-sonnet-20240620	0.964	0.680	0.810
Diagram	Yes	gemini-1.5-pro-exp-0827	0.961	0.858	0.953

Table A.14: **Comparing Gemini-1.5-Flash question categorization to larger models.** Gemini-1.5-Flash is evaluated against SOTA models on 1000 samples from VisionArena-Chat, using Gemini-1.5-Pro for image-based prompts and Claude-3.5-Sonnet for text-based prompts. Gemini-1.5-Flash achieves high agreement with SOTA models for category classification.

whether the given image contains a chart, diagram, or figure.

Carefully examine the user prompt and consider the following aspects:

1. Does the image contain visual elements such as graphs, flowcharts, tables, method figures, chemical structures, or other visual representations of data or concepts?
2. Does the prompt require interpreting or analyzing the flow of information, relationships between elements, or the structure of the visual representation in the image?
3. Does the prompt require spatial reasoning and understanding the layout or structure of the visual elements?
4. Does the image contain only text, tables, handwriting, or photographs without any visual representations of data or concepts? If so, it is NOT considered a chart or diagram.

Output your verdict in the following format:

"<decision>[yes/no]<decision>". Do NOT explain.

Creative Writing System Prompt (prompt only) You are tasked with determining whether a given VQA user prompt is asking for creative writing. Creative writing is defined as any form of writing that goes beyond standard professional, journalistic, academic, or technical literature. It typically involves imagination, originality, and expression of thoughts and emotions. Prompts which only ask to caption

the image without any other requests do NOT count as creative writing. Creative writing can include, but is not limited to, the following formats:

- Fiction (e.g., short stories, novels),
- Poetry (e.g., sonnets, free verse),
- Dramatic writing (e.g., screenplays, monologues, scripts),
- Personal essays (focusing on subjective experiences or narrative storytelling),
- Songs and lyrics

Carefully analyze the user prompt and consider whether it primarily requires creative writing. Think about the following aspects:

1. Does the prompt ask for fictional content, speculative scenarios, or the use of imagination to construct narratives?
2. Does it encourage the expression of thoughts, emotions, or personal experiences beyond mere factual reporting or analysis?
3. Is it asking for writing in a specific creative format (e.g., story, poem, script, etc)?
4. Is the primary purpose of the prompt to foster creative expression or originality rather than information delivery, technical documentation, or analytical reasoning?
5. Does the prompt request stylistic or rhetorical elements often associated with creative writing, such as metaphor, imagery, dialogue, etc?
6. Does the prompt expect a response in natural language (e.g., sentences, paragraphs) rather than visual, mathematical, or non-linguistic output?

Output your verdict in the following format:

"<decision>[yes/no]<decision>". Do NOT explain.

A.20 Contamination with Existing Benchmarks

To ensure that our results from [section 5.5](#) are not due to training on questions from the test sets, we investigate the rate of benchmark contamination in VisionArena-Chat. Using OpenAI’s text-embedding-small embeddings, we compute the cosine similarity between each VisionArena-Chat question and all benchmark questions, selecting the nearest neighbor with the highest similarity

score. We then count the number of cases where this similarity is ≥ 0.8 , indicating minor rephrasings of the same question. Table A.15 shows that less than 2% of benchmark questions are seen on VisionArena-Chat.

Dataset	# Matches	% dataset	% VisionArena-Chat
MMMU	47	0.4%	0.02%
MME	0	0.0%	0.0%
HallusionBench	0	0.0%	0.00%
MMMU Pro	23	1.3%	0.01%

Table A.15: ~~Proportion of benchmark data in VisionArena-Chat.~~

A.21 VisionArena-Bench

VLM-as-a-Judge System Prompt Please act as an impartial judge and evaluate the quality of the responses provided by two AI assistants to the user prompt displayed below. You will be given assistant A’s answer and assistant B’s answer. Your job is to evaluate which assistant’s answer is better.

Begin your evaluation by generating your own answer to the prompt. You must provide your answers before judging any answers.

When evaluating the assistants’ answers, compare both assistants’ answers with your answer. You must identify and correct any mistakes or inaccurate information.

Then consider if the assistant’s answers are helpful, relevant, and concise. Helpful means the answer correctly responds to the prompt or follows the instructions. Note when user prompt has any ambiguity or more than one interpretation, it is more helpful and appropriate to ask for clarifications or more information from the user than providing an answer based on assumptions. Relevant means all parts of the response closely connect or are appropriate to what is being asked. Concise means the response is clear and not verbose or excessive.

Then consider the creativity and novelty of the assistant’s answers when needed. Finally, identify any missing important information in the assistants’ answers that would be beneficial to include when responding to the user prompt.

After providing your explanation, you must output only one of the following choices as your final verdict with a label: 1. Assistant A is significantly better: $[[A \gg B]]$

2. Assistant A is slightly better: $[[A > B]]$

3. Tie, relatively the same: $[[A = B]]$

4. Assistant B is slightly better: $[[B > A]]$

5. Assistant B is significantly better: $[[B \gg A]]$

Example output: "My final verdict is tie: $[[A = B]]$ ".

Model	Score	95% CI	Token #
gpt-4o-2024-05-13	67.7	(-1.7, 1.8)	316
gemini-1.5-pro-exp-0827	66.2	(-1.8, 1.5)	329
gemini-1.5-flash-exp-0827	60.3	(-1.9, 1.9)	367
claude-3.5-sonnet-20240620	54.5	(-2.1, 1.9)	262
gpt-4-turbo-2024-04-09	50.0	(0.0, 0.0)	258
gemini-1.5-pro-001	45.5	(-1.8, 2.0)	261
gpt-4o-mini-2024-07-18	40.0	(-2.3, 1.9)	224
gemini-1.5-flash-8b-exp-0827	30.6	(-2.3, 1.8)	354
internvl2-26b	23.3	(-2.1, 1.1)	515
gemini-1.5-flash-001	23.0	(-1.1, 1.6)	271
claude-3-opus-20240229	18.9	(-1.9, 1.7)	201
claude-3-sonnet-20240229	18.4	(-1.4, 1.3)	205
reka-core-20240501	15.6	(-1.3, 1.4)	252
llama-3.2-11b-vision-instruct	11.2	(-1.3, 1.1)	457
claude-3-haiku-20240307	9.6	(-1.1, 1.0)	155
internvl2-4b	6.8	(-0.9, 0.8)	421

Table A.16: VisionArena-Bench leaderboard (baseline: GPT-4-Turbo)

Model	Score	95% CI	Token #
gemini-1.5-pro-exp-0827	87.6	(-1.1, 1.1)	329
gpt-4o-2024-05-13	86.8	(-1.1, 1.3)	316
claude-3-5-sonnet-20240620	86.3	(-1.3, 1.2)	262
gemini-1.5-flash-exp-0827	83.5	(-1.7, 1.1)	367
gpt-4-turbo-2024-04-09	80.7	(-1.0, 1.6)	258
gemini-1.5-pro-001	75.3	(-1.7, 1.4)	261
gpt-4o-mini-2024-07-18	73.0	(-1.3, 1.4)	224
gemini-1.5-flash-8b-exp-0827	64.7	(-1.5, 2.4)	354
gemini-1.5-flash-001	58.4	(-1.9, 1.5)	271
internvl2-26b	54.2	(-1.7, 1.7)	515
claude-3-opus-20240229	52.0	(-2.0, 1.7)	201
claude-3-sonnet-20240229	50.0	(0.0, 0.0)	205
reka-core-20240501	37.9	(-1.9, 1.7)	252
llama-3.2-11b-vision-instruct	32.6	(-1.7, 1.8)	457
claude-3-haiku-20240307	30.7	(-2.3, 1.6)	155
internvl2-4b	19.6	(-1.9, 1.3)	421

Table A.17: VisionArena-Bench leaderboard (baseline: claude-3-sonnet-20240229)

A.22 Additional model details

Table A.18 shows the mapping from the model names used in Section 5.4 to the exact model versions.

A.23 Failure Cases

Hard OCR (Figure A.31). While VLMs perform well at transcribing easily legible text, they struggle with perturbed text (e.g., rotations, blur). Reading such difficult text is essential for real-world applications. We show two failure cases: one with unclear handwriting and another with rotated text.

Counting (Figure A.32). Counting is a critical skill for decision-making across education, organization, and daily life. While humans count effortlessly, VLMs still struggle. We provide two examples where the top three models fail: one requiring counting based on shape and color, and another involving intersecting triangles.

Reasoning (Figure A.33, Figure A.34). Reasoning is essential for helping users tackle complex problems. While reasoning remains a challenge for both VLMs and LLMs, we present five unique VLM failure cases.

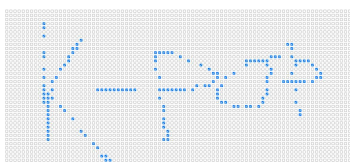
Model Version	Model Name
claude-3-5-sonnet-20240620	Claude 3.5 Sonnet
claude-3-haiku-20240307	Claude 3 Haiku
claude-3-opus-20240229	Claude 3 Opus
claude-3-sonnet-20240229	Claude 3 Sonnet
cogvlm2-llama3-chat-19b	CogVLM2 Llama3 Chat 19b
gemini-1.5-flash-api-0514	Gemini 1.5 Flash
gemini-1.5-pro-api-0514	Gemini 1.5 Pro
gemini-1.5-pro-exp-0801	Gemini 1.5 Pro Exp
gpt-4-turbo-2024-04-09	GPT-4 Turbo
gpt-4o-2024-05-13	GPT-4o
gpt-4o-mini-2024-07-18	GPT-4o Mini
internvl2-26b	InternVL2 26b
llava-v1.6-34b	LLAVA 1.6 34b
minicpm-v-2.6	MiniCPM v2.6
phi-3-vision-128k-instruct	Phi 3 Vision 128k Instruct
reka-core-20240501	Reka Core
reka-flash-preview-20240611	Reka Flash Preview

Table A.18: Model Name to exact model version

Model	K-Pop	Sign	Shapes	Triangles	Meme	Map	Shoes	Chess	ARC
gemini-1.5-pro-exp-0827	X	X	X	X	X	X	X	X	X
gpt-4o-2024-05-13	X	O	X	X	X	X	O	X	X
claude-3-5-sonnet-20240620	X	X	X	X	X	X	X	X	X
claude-3-opus-20240229	X	X	X	X	X	X	O	X	X
gpt-4-turbo-2024-04-09	X	X	X	X	X	X	X	X	X
gpt-4o-mini-2024-07-18	X	X	X	X	X	X	X	X	X
gemini-1.5-pro-001	X	X	X	X	X	X	X	X	X
gemini-1.5-flash-8b-exp-0827	X	X	X	X	X	X	X	X	X
gemini-1.5-flash-exp-0827	X	O	O	O	X	X	X	X	X
internvl2-26b	X	X	X	X	X	X	X	X	X
gemini-1.5-flash-001	X	X	O	X	X	X	X	X	X
claude-3-sonnet-20240229	X	X	X	X	X	X	O	X	X
llama-3.2-11b-vision-instruct	X	O	X	O	X	X	X	X	X
claude-3-haiku-20240307	X	X	X	X	X	X	X	X	X
internvl2-4b	X	X	X	X	X	X	X	X	X

Table A.19: **Model performance across several hard tasks.** O indicates that the model solves the problem and X indicates that the model fails to solve the problem. 9 out of 16 models fail all questions.

What does it put here?



Answer: K-Pop

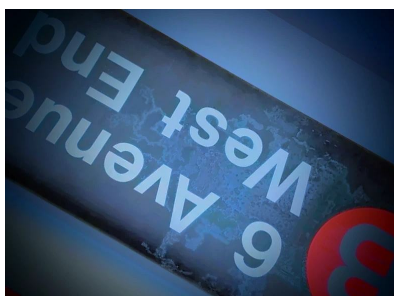
gemini-1.5-pro-exp-0827: It puts the word "HELLO" in a grid using blue checkmarks.

gpt-4o-2024-05-13: The image appears to be a **Minesweeper game board**. The blue numbers indicate the number of mines adjacent to each square. The numbers visible in the image are: - 1 - 2 - 3 - 4 These numbers help the player deduce where the mines are located on the board. The goal of the game is to clear all the non-mine squares without detonating any mines.

claude-3-5-sonnet-20240620: The image shows a grid of small squares, with some squares filled in blue to form letters. The blue squares spell out "THE LOTION" in a pixelated or dot-matrix style font across the grid.

(a) K-Pop Failure Case

What is written?



Answer: 6 Avenue West End

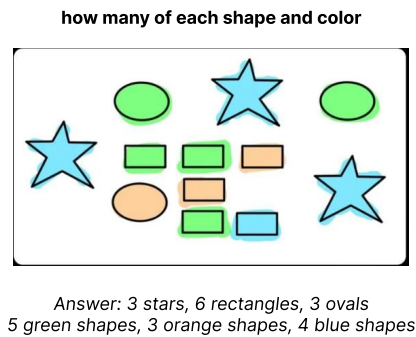
gemini-1.5-pro-exp-0827: 9
West End Avenue

gpt-4o-2024-05-13: The text on the sign reads "6 Avenue West End."

claude-3-5-sonnet-20240620: The image shows text that reads "West 6 Avenue" when read from right to left. The text appears to be on a street sign or similar type of signage. The lettering is white on a dark blue background, and the image has a slightly blurred or distorted quality to it, possibly due to the angle or lighting conditions when the photo was taken.

(b) Sign Failure Case

Figure A.31: **Hard OCR Failure Cases.** These failure cases show that models still fail in cases where the text is perturbed such as rotations or messy hand-writing.



gemini-1.5-pro-exp-0827: Here's a count of the shapes and colors:

****Shapes:****
****Stars:** 3**
****Ovals:** 3**
****Rectangles:** 5**
****Colors:****
****Light Blue:** 4 (3 stars, 1 rectangle)**
****Light Green:** 4 (2 ovals, 2 rectangles)**
****Light Orange/Peach:** 2 (1 oval, 1 rectangle)**
 Let me know if you have any other questions!

gpt-4o-2024-05-13: In the image, there are various shapes and colors. Here is the count for each shape and color:

****Shapes:****
- Stars: 4
- Ovals: 3
- Rectangles: 6
****Colors:****
- Blue: 5 (4 stars, 1 rectangle)
- Green: 5 (3 ovals, 2 rectangles)
- Orange: 3 (1 oval, 2 rectangles)

claude-3-5-sonnet-20240620: Here's a breakdown of the shapes and colors in the image:

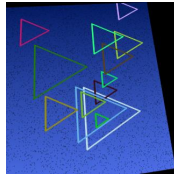
Shapes:
- Stars: 3 (all light blue)
- Circles/Ovals: 3 (2 green, 1 peach/light orange)
- Rectangles: 5 (3 green, 1 light blue, 1 peach/light orange)

Colors:
- Light Blue: 4 shapes (3 stars, 1 rectangle)
- Green: 5 shapes (2 circles, 3 rectangles)
- Peach/Light Orange: 2 shapes (1 oval, 1 rectangle)

In total, there are 11 shapes in the image.

(a) Shapes Failure Case

How many distinct triangles there are on the image? Intersections between figures dont count.



Answer: 13

gemini-1.5-pro-exp-0827: Let's count the distinct triangles in the image: 1. Top left: 1 (pink) 2. Top right: 1 (light purple) 3. Middle left: 1 (green) 4. Center: - Largest: 1 (light blue) - Medium: 1 (white) - Smallest: 1 (bright green) - Overlapping: 1 (brown) - Tiny: 1 (dark red) 5. Right: 1 (light blue)

Total count: 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = 9

There are ****9**** distinct triangles in the image.

gpt-4o-2024-05-13: There are **11** distinct triangles in the image.

claude-3-5-sonnet-20240620: In the image, there are **10** distinct triangles. These triangles are of various sizes and colors, scattered across a blue background. The triangles appear in different colors including white, green, pink, yellow, and light blue. Some of the triangles are overlapping, but as per your instruction, we're not counting intersections between figures. Each triangle is considered as a separate, distinct shape regardless of its position or overlap with others.

(b) Triangles Failure Case

Figure A.32: **Hard Counting Failure Cases.** These cases show that proprietary models still fail at counting tasks involving shapes even when the format is clear.

Which line connects Trolskaholm to Solköping?



Answer: The Ferry Line

gemi-1.5-pro-exp-0827: The ****Sunrise**** line connects Trolskaholm to Solköping.

It's the yellow/gold line on the map. **You can follow it from Trolskaholm all the way down to Solköping.**

gpt-4o-2024-05-13: The line that connects **Trolskaholm to Solköping** is the **"Midnight" line**, which is represented by the blue color on the map.

claude-3-5-sonnet-20240620: The line that connects Trolskaholm to Solköping is the Bullet line, which is represented by the red line on the map. **This red line runs from Trolskaholm through several stations including Alfheim Central and Singer Park**, before reaching Solköping in the southern part of the map.

(a) Map Failure Case

gemi-1.5-pro-exp-0827: The image is a humorous meme playing on the mathematical concept of square roots.

****The Setup:****

- * It shows two pictures of the same cat from behind.
- * The cat has a distinctive black marking on its back that vaguely resembles a square root symbol ($\sqrt{}$).

****The Joke:****

- * The first image is labeled "Cat".
- * The second image is labeled " $\sqrt{\text{Cat}}$ ".

****The Meaning:****

The joke lies in the visual pun. **The cat's marking looks like a square root symbol**, so the second image humorously implies that the cat has been "square rooted" – as if it were a mathematical equation. It's a silly and absurd concept, which is what makes it funny.

gpt-4o-2024-05-13: This image is a humorous visual pun involving a cat.

On the left side, there is a picture of a cat sitting on the ground, viewed from behind. Above this image, the word "Cat" is written.

On the right side, there is a similar picture of the same cat, but the image has been edited to make the cat appear wider and shorter. Above this image, the text " $\sqrt{\text{Cat}}$ " (square root of cat) is written.

The joke is that **the cat on the right looks like it has been "squared" (increased in width and decreased in height)**, playing on the mathematical concept of squaring and square roots. The cat on the right is meant to humorously represent the "square root" of the cat on the left.

claude-3-5-sonnet-20240620: This image is a humorous play on words and mathematics, using pictures of a cat to illustrate the concept.

The top of the image shows two text labels:

1. "Cat" on the left
2. " $\sqrt{\text{Cat}}$ " (square root of Cat) on the right

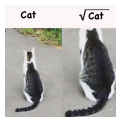
Below these labels are two photos of the same cat, presumably:

1. On the left, we see a normal-sized cat sitting on a paved surface. It has a black and white coat.
2. On the right, we see what appears to be the same cat, but it looks elongated or stretched vertically. This creates the illusion that the cat is taller or longer than in the left image.

The joke here is that the **"square root" of Cat ($\sqrt{\text{Cat}}$) is visually represented by a stretched version of the cat, as if taking the square root has "elongated" the animal. It's a clever visual pun combining mathematical notation with a literal interpretation using cat photos.**

This type of humor often resonates well on social media platforms, combining familiar concepts (cats and basic math) in an unexpected and visually amusing way.

Explain this image.



Answer: The inside fur of the cat looks like another cat, so the cat's square root is its own cat-looking fur.

(b) Meme Failure Case

The photo shows a boy's new white sneaker and a girl's black leather shoe. Can you describe what happened?



Answer: The girl with the black leather shoe stepped on the boy with the white sneaker.

gemi-1.5-pro-exp-0827: Based on the photo, **it appears the boy with the white sneaker has stepped on the girl's black leather shoe**. The white sneaker is positioned over the black shoe, suggesting that it has come down on top of it. This could have happened accidentally in a crowded space, or perhaps during play or a moment of clumsiness. Without more context, it's impossible to know the exact circumstances, but the positioning of the shoes strongly indicates that the white sneaker has stepped on the black shoe.

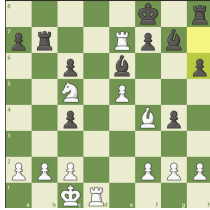
gpt-4o-2024-05-13: The photo shows **a black leather shoe stepping on a white sneaker**. It appears that the girl wearing the black leather shoe has accidentally or intentionally stepped on the boy's new white sneaker. This could have resulted in the white sneaker getting scuffed or dirty.

claude-3-5-sonnet-20240620: Based on the image, it appears that the white sneaker and the black leather shoe have come into contact with each other. The white sneaker looks new and clean, while the **black leather shoe seems to be pressing against or stepping on the back part of the white sneaker**. This situation often occurs when someone accidentally steps on the back of another person's shoe while walking closely behind them. It can lead to the heel of the shoe being pushed down or the back of the shoe getting scuffed. The contrast between the pristine white sneaker and the darker leather shoe is quite noticeable in the image. This kind of incident can be frustrating, especially with new shoes, as it might leave marks or damage the shoe's appearance. It's a common mishap that can happen in crowded areas or when people are walking in close proximity to each other.

(c) Shoes Failure Case

Figure A.33: **Hard Reasoning Failure Cases (part 1).** The failure cases highlight the model's inability to connect the visual reasoning with language reasoning. While the vision model identifies an object, it is not able to identify the relationship between them correctly.

I'm playing as white, what should I do in this position?? I'm confused.



Possible Answer: Re7 Rb7 to take the Rook

gemi-1.5-pro-exp-0827: The best move for white in this position is

Rxe7+

Explanation

1. ****Rxe7+**** forces the black king to capture the rook (Kxe7).
2. ****Qxg7+**** follows, checking the black king again.
3. The black king is forced to move to h8 (Kh8).
4. ****Qxh8#**** delivers checkmate.

This is a forced checkmate sequence that wins the game for white.

* Moving pawns on the queenside doesn't achieve anything significant and allows black to consolidate their position.

* Developing the knight to f3 doesn't pose an immediate threat and black can easily defend.

* Other rook moves don't lead to a forced checkmate.

Therefore, Rxe7+ is the only winning move for white in this position.

gpt-4o-2024-05-13: In this position, you have a strong attacking opportunity as White. You can deliver a checkmate in two moves.

Here's what you should do:

1. ****Rd8+**** (Rook to d8 check)

- This move forces the Black king to move to e7 because the rook on d8 is **protected by the knight on c5**.

2. ****Re8#**** (Rook to e8 checkmate)

- The rook on e8 delivers checkmate as the Black king has no legal moves and cannot escape the check.

So, the sequence is:

1. Rd8+ Ke7

2. Re8#

This results in a checkmate.

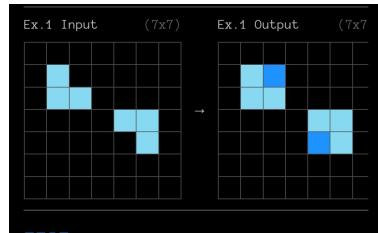
claude-3-5-sonnet-20240620: In this chess position, you're playing as White and have a few good options to consider. Here are some suggestions:

1. The most immediate threat is **Black's bishop on f6**, which is attacking **your rook on d4**. You should address this threat first.
2. One strong move would be to capture the **black pawn on e7** with your rook: **Rxe7+**. This move puts Black's king in check and removes the defender of the f6 bishop.
3. After Black responds to the check (likely by moving the king to f8), you can then capture the bishop on f6 with your bishop: **Bxf6**.
4. This sequence would result in you winning a pawn and a bishop for the exchange of one of your rooks, which is generally a good trade.
5. Another option, if you want to play more cautiously, is to simply move your rook away from the attack, perhaps to d3 or d2.
6. You could also consider developing your pieces further, such as moving your knight on c5 to a more active square like e6, which would attack Black's rook on b7.

Remember, chess is about long-term strategy as well as immediate tactics. Consider how your move will affect the overall position and try to improve your piece placement while addressing immediate threats.

(a) Chess Failure Case

Convert the image into a text based format for both the input and output image, use "[]" to represent each square in a grid, "[b]" to represent light blue grids and "[B]" to represent dark blue grids



Answer:

[] [] [] [] [] [] []	[] [] [] [] [] [] []
[] [b] [] [] [] [] []	[] [b] [B] [] [] [] []
[] [b] [b] [] [] [] []	[] [b] [b] [] [] [] []
[] [] [] [b] [] [] []	[] [] [] [b] [b] [] []
[] [] [] [] [b] [] []	[] [] [] [] [B] [b] []
[] [] [] [] [] [] []	[] [] [] [] [] [] []
[] [] [] [] [] [] []	[] [] [] [] [] [] []

gemi-1.5-pro-exp-0827

Input:

```
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [b] [ ] [ ] [ ] [ ] [ ]
[ ] [b] [b] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [b] [b] [ ] [ ]
[ ] [ ] [ ] [ ] [b] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

Output:

```
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [b] [ ] [ ] [ ] [ ] [ ]
[ ] [b] [b] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [B] [b] [ ] [ ]
[ ] [ ] [ ] [b] [b] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

gpt-4o-2024-05-13

Ex.1 Input (7x7)

```
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [b] [b] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [b] [b] [ ] [ ]
[ ] [ ] [ ] [ ] [b] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

Ex.1 Output (7x7)

```
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [b] [B] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [b] [b] [ ] [ ]
[ ] [ ] [ ] [ ] [b] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

claude-3-5-sonnet-20240620

Input (7x7):

```
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [b] [ ] [ ] [ ] [ ] [ ]
[ ] [b] [b] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [b] [b] [ ] [ ]
[ ] [ ] [ ] [ ] [b] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

Output (7x7):

```
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [b] [B] [ ] [ ] [ ] [ ]
[ ] [b] [b] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [b] [b] [ ] [ ]
[ ] [ ] [ ] [ ] [b] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
[ ] [ ] [ ] [ ] [ ] [ ] [ ]
```

(b) ARC Failure Case

Figure A.34: **Hard Reasoning Failure Cases (part 2)**. These failure cases highlight the inability for the model to be able to correctly map out a grid-like structure and the various pieces in it.

Chapter 6: VibeCheck

A.24 Experimental Details & Dataset Statistics

Dataset	# Train	# Test
Human VS ChatGPT	250	250
Chatbot Arena - All	839	839
Chatbot Arena - STEM	346	347
Chatbot Arena - Writing	278	277
CNN/DailyMail	444	346
MATH	218	218
COCO w/ ShareGPT-4V Captions	323	346

Table A.20: Dataset Statistics

Dataset	Model A	Model B	Model A Win Rate
Human VS ChatGPT	Humans	GPT-3.5	-
Chatbot Arena - All	Llama3-70b-Instruct	GPT-4 + Claude-3-Opus	50%
Chatbot Arena - STEM	Llama3-70b-Instruct	GPT-4 + Claude-3-Opus	44%
Chatbot Arena - Writing	Llama3-70b-Instruct	GPT-4 + Claude-3-Opus	57%
CNN/DailyMail	Cohere Command X	TNLGv2	71.12%
MATH	GPT-4o	Llama3-405b	76%
COCO w/ ShareGPT-4V Captions	GPT-4V	Gemini-1.5-Flash	80%

Table A.21: Model Win Rates

num_eval_vibes = number of vibes to validate at every iteration

d = number of prompt output triples to use in each iteration of the vibe discovery phase

batch = number of triples to feed into the prompt of the discovery LLM at once.

iterations = number of vibe iterations to perform

num_final_vibes = number of vibes to evaluate at the end of all the iterations. This can be set to false, in which case all the vibes collected in the iteration

We take the 1000 captions generated by GPT-4V from the ShareGPT-4V dataset [33] and generate captions for the same images using the same captioning prompt using Gemini-1.5-Flash.

Dataset	d	batch	num_eval_vibes	num_final_vibes	iterations
Human VS ChatGPT	40	5	10	10	3
Chatbot Arena - All	20	5	10	10	3
Chatbot Arena - STEM	20	5	10	10	3
Chatbot Arena - Writing	20	5	10	10	3
CNN/DailyMail	20	2	10	10	3
MATH	20	5	10	10	1
COCO	20	5	10	10	1

Table A.22: **VibeCheck** Hyperparameters

A.25 Gold standard labels

Below is a summary of key differences found by human evaluators in the HC3 dataset [95] listed in their paper.

Characteristics of ChatGPT

- (a) Responses are well-organized, often starting with a definition of key concepts before providing a step-by-step explanation and concluding with a summary.
- (b) Answers tend to be detailed and extensive.
- (c) ChatGPT generally minimizes bias and avoids generating harmful content.
- (d) It refrains from responding to queries beyond its scope of knowledge.
- (e) In some cases, it may generate incorrect or fabricated information.

Differences Between Human and ChatGPT Responses

- (a) ChatGPT remains strictly on topic, while human responses may shift toward related or tangential subjects.
- (b) It tends to provide objective, fact-based answers, whereas human responses often include personal opinions or subjective elements.
- (c) ChatGPT’s tone is typically formal and structured, while human speech is more conversational and informal.
- (d) Unlike humans, ChatGPT does not express emotions, relying solely on linguistic structure rather than emotional cues like punctuation or tone variations.

A.26 Generating Preset Vibes

We generate our list of 10 preset vibes by prompting GPT-4o with the following:

Preset Vibe Generation Prompt

```
I am a machine learning researcher trying to figure out the major
differences between the behavior of different large language models.
Can you list common ways in which two language models can differ in
their outputs?

Please output a list differences between these sets of outputs with
relation to specific axes of variation. Try to give axes that a human
could easily interpret and they could understand what it means to be
higher or lower on that specific axis. Please ensure that the concepts
used to explain what is high and low on the axis are distinct and
mutually exclusive such that given any tuple of text outputs, a human
```


Vibe	Axis Definition (low → high)
Assertiveness	Uses tentative or uncertain language. → Uses definitive, confident statements.
Detail & Elaboration	Gives brief or shallow responses. → Provides thorough, nuanced, and expansive information.
Formality	casual, conversational, or informal language. → formal, sophisticated language and sentence structure.
Emotional Tone	Remains neutral or detached. → Infuses responses with expressive emotion and enthusiastic or empathetic tone.
Creativity & Originality	Sticks to standard, predictable answers. → Provides responses with novel ideas or imaginative scenarios.
Explicitness	Uses vague or implicit language. → States things directly and unambiguously.
Humor and Playfulness	Responds in a straightforward and serious manner. → Uses humor, playful language, or wordplay.
Engagement	Presents information passively. → Actively engages the reader using rhetorical questions or interactive phrasing.
Logical Rigor	Provides conclusions without thorough justification. → Constructs well-supported arguments with clear reasoning.
Conciseness	Uses verbose language and excessive details. → Uses minimal words to convey a point clearly.

Table A.23: **Predefined vibes.** We prompt GPT-4o to generate a set of 10 vibes which represent common axes on which LLM outputs differ.

could easily and reliably determine which model is higher or lower on that axis.

The format should be

- {axis 1}: {difference}
- {axis 2}: {difference}

Please output differences which have a possibility of showing up in future unseen data and which would be useful for a human to know about when deciding with LLM to use. For each axis, define clearly and succinctly what constitutes a high or low score, ensuring these definitions are mutually exclusive. Please give 10 differences

A.27 Additional VibeCheck Details

Vibe Discovery

Below is the user prompt we use for vibe discovery.

Vibe Discovery Prompt

The following are the results of asking a set language models to generate an answer for the same questions:

[PROMPT] [OUTPUT 1] [OUTPUT 2]

I am a machine learning researcher trying to figure out the major differences between these two LLM outputs so I can better compare the behavior of these models. Are there any variations you notice in the outputs?

Please output a list differences between these sets of outputs with relation to specific axes of variation. Try to give axes that a human could easily interpret and they could understand what it means to be higher or lower on that specific axis. Please ensure that the concepts used to explain what is high and low on the axis are distinct and mutually exclusive such that given any tuple of text outputs, a human could easily and reliably determine which model is higher or lower on that axis.

The format should be: `{{axis}}: Low: {{low description}}; High: {{high description}}`

Vibe Summarization. To summarize the set of vibes found in the vibe discovery process, We cluster the axes using agglomerative clustering on the embeddings of the axes generated by the 'hkunlp/instructor-xl' model, and prompt GPT-4o to reduce this set by removing any vibes which are similar. After this stage we are left with a set of less than 20 vibes which we use to score the outputs of each model.

Vibe Reduction Prompt

Below is a list of axes with a description of what makes a piece of text low or high on this axis. Are there any axes that have similar meanings based off their low and high descriptions? Are there any sets of axes that would convey the same information to a user (e.g. level of detail)? Could any of the low and high descriptions be simplified to make them easier to understand?

Please remove any axes with roughly the same meaning and simplify the descriptions of what makes a piece of text low or high on this axis. Please ensure that the descriptions of what makes a piece of text low

or high on this axis are distinct, useful, and mutually exclusive. Given any piece of text, a human should be able to easily and reliably determine if this text falls high or low on each axis.

Here is the list of axes: {axes}

Please return the simplified list of axes and the descriptions of what makes a piece of text low or high on this axis. These axes should contain only one concept and should be human interpretable. Some examples of bad axes include:

- "Configuration Clarity: High: Clearly defined structure and purpose. Low: Vaguely defined, minimal purpose." -> This axes is bad because it is not clear what a clearly defined purpose means nor what a vaguely defined purpose means.
- "Language and Communication: High: Varied/precise, complex structure. Low: Straightforward, simple or general language." -> This axes is bad because it combines multiple concepts into one axis.
- "Content Quality: High: High quality, engaging, informative. Low: Low quality, unengaging, uninformative." -> This axes is bad because it is not clear what high quality means nor what low quality means.

Some examples of good axes include:

- "Complexity: High: Complex, multi-layered, intricate. Low: Simple, straightforward, easy to understand."
- "Efficiency (coding): High: Code optimized for runtime, minimal memory usage. Low: Code inefficient, high memory usage."

Some examples of axes which should be combined include:

- "Emotional Tone: High: Contains emotionally charged language. Low: Maintains a neutral tone." and "Empathy: High: Shows empathy. Low: Only factual answers without empathy." are redundant because they both measure the emotional content of the text. If two similar axes are found, keep the one that is more informative or more specific.

Please maintain the format of the original axes and return a list like [{"axis name}: High: {high description} Low: {low description}", ...]. I should be able to parse this output into a string using `ast.literal_eval`. If the original list does not contain any redundant axes, please return the original list.

If the number of vibes after the first reduction step is $> K$, we prompt GPT-4o to reduce the set further with the final reducer prompt.

Final Vibe Reducer Prompt

Below is a list of axes with a description of what makes a piece of text low or high on this axis. I would like to summarize this list to at most {number} representative axes.

Here is the list of axes: [VIBES]

These axes should contain only one concept and should be human interpretable. Some examples of bad axes include:

- "Configuration Clarity: High: Clearly defined structure and purpose. Low: Vaguely defined, minimal purpose." -> This axis is bad because it is not clear what a clearly defined purpose means nor what a vaguely defined purpose means.
- "Language and Communication: High: Varied/precise, complex structure. Low: Straightforward, simple or general language." -> This axis is bad because it combines multiple concepts into one axis.
- "Content Quality: High: High quality, engaging, informative. Low: Low quality, unengaging, uninformative." -> This axis is bad because it is not clear what high quality means nor what low quality means.

Some examples of good axes include:

- "Complexity: High: Complex, multi-layered, intricate. Low: Simple, straightforward, easy to understand."
- "Efficiency (coding): High: Code optimized for runtime, minimal memory usage. Low: Code inefficient, high memory usage."

Some examples of axes which should be combined include:

- "Emotional Tone: High: Contains emotionally charged language. Low: Maintains a neutral tone." and "Empathy: High: Shows empathy. Low: Only factual answers without empathy." are redundant because they both measure the emotional content of the text. If two similar axes are found, keep the one that is more informative or more specific.

Please return the simplified list of $\leq K$ axes with any redundant axes removed and the descriptions of what makes a piece of text low or high on this axis simplified. Are there any axes which convey roughly the same information? Are there any axes where almost all samples which score highly on one axis would also score highly on the other?

Please maintain the format of the original axes and return a numbered list. Each element should be structured as follows: "{axis name}: High: {high description} Low: {low description}"

Vibe Validation

Prompt for ranker judge

I want to compare the outputs of two language models (A and B) for the same prompt. I would like you to evaluate where each output falls on the following axis: [VIBE].

If you had to choose which output is higher on the axis, which would you choose? Here is the prompt and the outputs of A and B respectively:

[PROMPT] [OUTPUT A] [OUTPUT B]

Please respond with which model you think is higher on the axis and explain your reasoning. If this axis does not apply to these examples or these outputs are roughly equal on this axis, return "N/A".

Vibe Iteration

At iteration step t , we are left with k distinct vibes which are well-defined and differentiating along with their scores $\nu_{1:k}(p, o_A, o_B)$. Using these scores, we train a LR model to predict LLM identity (i.e. "Is the response shown first LLM A or LLM B?") and get the predictions on our entire set D . Assuming we have not hit the max iteration steps set by the user, we iterate if the number of samples misclassified by the model matching predictor is greater than the number of prompts to perform discovery on (d). In iteration step $t + 1$, we take these misclassified prompt output triples in batches of size *batch* along with the current set of vibes $\nu_1, \dots, \nu_k$ and prompt the LLM to generate new differences between outputs what are not represented in the current vibes. These vibes are then reduced using the same procedure as the vibe discovery process. In practice we found that often some of the reduced vibes from the discovery phase at $t + 1$ were redundant with an existing axis, so we perform one more deduplication step using the prompt below.

Vibe Discovery Iteration step

Given a new set of responses, your task is to expand on the set of axes which have been previously identified by finding other clear differences between the responses that are not captured by the existing axes. The expanded axes should be any differences between responses that are not clearly captured by the existing axes. Be as exhaustive as possible in listing differences on as many different axes as you can think of, and be specific about what constitutes high and low on each axis.

Your axis should be interpretable: a human should easily and reliably determine which response is higher, lower, or even on this axis when given a new set of responses. Please do not make your axes too broad and list as many axes as you can think of that are not covered by the existing axes. Most of these new axes should be either completely different from the existing axes or should highlight a more finegrained difference which an existing axis might broadly cover. For instance,

if an existing axis is "Enthusiasm: High: enthusiastic, Low: unenthusiastic", a new axis might be "Use of Exclamation Points", or if an existing axis is "Cultural Context: High: culturally relevant, Low: culturally irrelevant", a new axis might be "Use of Slang". Please think through the axes carefully and make sure they are clear, concise, and do not overlap with each other or the existing axes. Do not include any of the existing axes in your response. Your output should be in this format:

New Axes:

- {axis 1}:

High: {description of high}

Low: {description of low}

- {axis 2}:

High: {description of high}

Low: {description of low}

Do not include any other information in your response.

Vibe deduplication in iteration step $t + 1$

Here is a list of axes on which two strings may vary. Each axis has a description of what makes a string high or low on that axis.

[EXISTING AXES]

[NEW AXES]

It is likely that several of these axes measure similar things. Your task is to remove any redundant axes. Think about if a user would gain any new information from seeing both axes. For example, "Emotional Tone: High: Contains emotionally charged language. Low: Maintains a neutral tone." and "Empathy: High: Shows empathy. Low: Only factual answers without empathy." are redundant because they both measure the emotional content of the text. If two similar axes are found, keep the one that is more informative.

Output the reduced list of axes, separated by a newline. All of the axes should maintain the same format they have in the list of {axis}:

High: {high} Low: {low}

Generating Preference Labels

prompt for generating preference labels

Please act as an impartial judge and evaluate the quality of the responses provided by two AI assistants (A and B) to the user question

displayed below. You should choose the assistant that follows the user’s instructions and answers the user’s question better. Your evaluation should consider factors such as the helpfulness, relevance, accuracy, depth, creativity, and level of detail of their responses. Begin your evaluation by comparing the two responses and provide a short explanation. Avoid any position biases and ensure that the order in which the responses were presented does not influence your decision. Do not allow the length of the responses to influence your evaluation. Do not favor certain names of the assistants. Be as objective as possible.

Here is the prompt and the outputs of A and B respectively:

[PROMPT] [OUTPUT A] [OUTPUT B]

Please respond with the model which contains a higher quality response. Based on your analysis, please explain your reasoning before assigning a score. Use the following format for your response:

Analysis: {reasoning}

Model: {A, B, tie}

A.28 Further Related Works

Automatic metrics for benchmark evaluations. The number of benchmarks in the NLP community has exploded in recent years, with a wealth of work on providing a more holistic evaluation of language models beyond just accuracy. Several works [13, 200, 234], aim to improve on automatic metrics like BLEU [201] and ROUGE [156] scores to better measure how well a models output aligns with the ground truth by incorporating more nuanced evaluation criteria like factual accuracy, fluency, and conciseness. Similarly, efforts have been made [153] to standardize model evaluation by evaluating models on many of these metrics.

A.29 Limitations

Prompt: The following equation describes a parabola. If "a" is negative, which way does the parabola open? $x = ay^2$

Llama: A nice simple question! ... The correct answer is down. **GPT:** If $\backslash(a)$ is negative, the parabola opens to the left.

Technical Accuracy and Terminology: Factually correct, precise terminology → Factual inaccuracies, vague terminology.
Ranker Decision: Llama **Correct Answer:** Left (GPT's answer)

(a) Weaknesses in the mathematical abilities of the LLM judge (GPT-4o-mini).

A.30 Vibes from each Application

Vibe (low -> high)	Sep Score [-0.5,0.5]	PP Coef [-0.6,0.6]	Cohen
Conciseness. Elaborate and lengthy explanations. → Short and to the point.			0.37
Citation and References. Avoids citations, smoother text flow. → Includes references and citations for credibility.			0.41
Emotional Tone and Empathy. Clinical and straightforward, less emotional engagement. → Uses comforting language, acknowledges emotional challenges.			0.46
Technical Depth. Simplified, general, and basic technical explanations. → Detailed, formal, and multifaceted technical explanations.			0.65
Legal and Safety Considerations. Does not consistently include disclaimers. → Includes disclaimers or notes about advice limitations.			0.29
Contextual Information. Focuses strictly on the topic. → Provides additional irrelevant context and discussion.			0.39
Practical Advice and Safety. Addresses concerns directly, less emphasis on professional help. → Practical, cautious advice, emphasizes seeking professional help.			0.43
Detail Orientation. Concise and limited responses covering fewer aspects. → Thorough and comprehensive responses covering multiple aspects.			0.55
Response Length. Short, to-the-point responses. → Long, informative responses.			0.50
Formality and Tone. Casual, relaxed tone with conversational language. → Formal, academic tone throughout.			0.64

(a) Human VS ChatGPT outputs on HC3 [95]

Vibe (low -> high)	Sep Score [-0.4,0.4]	PP Coef [-0.5,0.5]	Cohen
Language and Tone. Professional, straightforward tone. → Enthusiastic, friendly tone.			0.51
Typographic Emphasis. Minimal use of typographic emphasis, letting the text stand alone. → Uses typographic emphasis like bold or italics to highlight key points.			0.64
Interactivity. Provides information passively without engaging the user. → Encourages user interaction, such as posing questions or suggesting actions.			0.44
Formatting Completeness. Responses are minimally formatted, relying on plain text. → Responses include comprehensive formatting, such as Markdown or additional stylistic elements.			0.57
Examples and Illustrations. Minimal examples. → Provides multiple examples.			0.61
Use of Humor. Maintains a serious tone without humorous elements. → Employs humor frequently to engage the reader.			0.62
Use of Personal Pronouns. Rarely or never uses personal pronouns. → Frequently uses personal pronouns (I, we, you).			0.32
Ethical Consideration. Provides factual information without commenting on ethics. → Offers ethical considerations in its responses.			0.53
Humility. Projects confidence and completeness without discussing limitations. → Frequently acknowledges limitations in the response or areas of uncertainty.			0.41
Formality Level. Uses informal or conversational language. → Uses formal language and expressions.			0.45

(a) VibeCheck vibes on Chatbot Arena[Overall]

Prompt: Is European Thyroid Journal only open access or mixed open access and subscription?

Llama: ...the European Thyroid Journal (ETJ) is a hybrid open-access journal, which means it offers both open-access and subscription-based models.

GPT: As of my last update in 2023, the European Thyroid Journal operates under an Open Access model.

Vibe: Technical Accuracy and Terminology: High: Factually correct with precise terminology. Low: Factual inaccuracies and vague terminology.

(b) The answer to certain questions changes depending on the following parameters:

- (1) When was the question asked?
- (2) What is the knowledge cutoff of Model A and Model B?
- (3) What is the knowledge cutoff of the LLM ranker ensemble?

These types of questions lead to unreliable ranker evaluations and reduced inter-annotator agreement.

Vibe (low -> high)	Sep Score [-0.3,0.3]	PP Coef [-0.7,0.7]	Cohen
Engagement. Presents information passively. → Actively engages the reader using rhetorical questions or interactive phrasing.			0.48
Emotional Tone. Remains neutral or detached. → Infuses responses with expressive emotion, making the tone enthusiastic or empathetic.			0.53
Humor and Playfulness. Responds in a straightforward and serious manner. → Uses humor, playful language, or wordplay to make the response engaging.			0.64
Creativity and Originality. Sticks to standard, predictable answers. → Provides responses with novel ideas or imaginative scenarios.			0.51
Detail and Elaboration. Gives brief or shallow responses. → Provides thorough, nuanced, and expansive information.			0.60
Assertiveness. Uses tentative or uncertain language. → Uses definitive, confident statements.			0.49
Explicitness. Uses vague or implicit language. → States things directly and unambiguously.			0.43
Logical Rigor. Provides conclusions without thorough justification. → Constructs well-supported arguments with clear reasoning.			0.48
Conciseness. Uses verbose language and excessive details. → Uses minimal words to convey a point clearly.			0.40
Formalness. Uses casual, conversational, or informal language. → Uses formal and sophisticated vocabulary and sentence structure.			0.50

(b) Preset vibes on Chatbot Arena[Overall]

Vibe (low -> high)	Sep Score [-0.2,0.2]	PP Coef [-0.8,0.8]	Cohen
Assertiveness. Uses tentative or uncertain language. → Uses definitive, confident statements.			0.34
Conciseness. Uses verbose language and excessive details. → Uses minimal words to convey a point clearly.			0.34
Creativity and Originality. Sticks to standard, predictable answers. → Provides responses with novel ideas or imaginative scenarios.			0.47
Detail and Elaboration. Gives brief or shallow responses. → Provides thorough, nuanced, and expansive information.			0.62
Emotional Tone. Remains neutral or detached. → Infuses responses with expressive emotion, making the tone enthusiastic or empathetic.			0.45
Engagement. Presents information passively. → Actively engages the reader using rhetorical questions or interactive phrasing.			0.35
Explicitness. Uses vague or implicit language. → States things directly and unambiguously.			0.36
Formalness. Uses casual, conversational, or informal language. → Uses formal and sophisticated vocabulary and sentence structure.			0.56
Humor and Playfulness. Responds in a straightforward and serious manner. → Uses humor, playful language, or wordplay to make the response engaging.			0.59
Logical Rigor. Provides conclusions without thorough justification. → Constructs well-supported arguments with clear reasoning.			0.45

(b) Preset vibes on Chatbot Arena[STEM]

Vibe (low -> high)	Sep Score [-0.3,0.3]	PP Coef [-0.5,0.5]	Cohen
Engagement and Enthusiasm. The response is more formal, neutral, and factual without engaging language. -> The response exudes enthusiasm and engages the reader, often employing exclamation points, a friendly tone, and casual conversational remarks.			0.43
Error Handling. Minimal or no error handling, assumes ideal scenarios. -> Includes comprehensive error handling and user input validation within the code.			0.33
Handling of Uncertain Information. States information definitively without disclaimers. -> Clearly indicates uncertainty or assumptions.			0.38
Interactivity and Engagement. Formal, direct tone focused on clarity. -> Engaging tone, tutorial-like.			0.44
Jargon and Terminology. Uses general language and avoids jargon. -> Uses specialized jargon and complex terms.			0.37
Safety and Accuracy Emphasis. Lacks explicit emphasis on safety or ethics. -> Includes disclaimers, emphasizes ethical considerations.			0.26
Tone and Enthusiasm. Neutral, utilitarian. -> Engaging, enthusiastic.			0.44

(a) VibeCheck vibes on Chatbot Arena [STEM]. Note that we only find 7 vibes which achieve a separability score on the training set about the 0.05 threshold.

Vibe (low -> high)	Sep Score [-0.4,0.4]	PP Coef [-0.6,0.6]	Cohen
Assertiveness. Uses tentative or uncertain language. -> Uses definitive, confident statements.			0.56
Conciseness. Uses verbose language and excessive details. -> Uses minimal words to convey a point clearly.			0.36
Creativity and Originality. Sticks to standard, predictable answers. -> Provides responses with novel ideas or imaginative scenarios.			0.46
Detail and Elaboration. Gives brief or shallow responses. -> Provides thorough, nuanced, and expensive information.			0.64
Emotional Tone. Remains neutral or detached. -> Infuses responses with expressive emotion, making the tone enthusiastic or empathetic.			0.55
Engagement. Presents information passively. -> Actively engages the reader using rhetorical questions or interactive phrasing.			0.55
Explicitness. Uses vague or implicit language. -> States things directly and unambiguously.			0.41
Formalness. Uses casual, conversational, or informal language. -> Uses formal and sophisticated vocabulary and sentence structure.			0.60
Humor and Playfulness. Responds in a straightforward and serious manner. -> Uses humor, playful language, or wordplay to make the response engaging.			0.61
Logical Rigor. Provides conclusions without thorough justification. -> Constructs well-supported arguments with clear reasoning.			0.45

(b) Preset vibes on Chatbot Arena [Writing]

Vibe (low -> high)	Sep Score [-0.4,0.4]	PP Coef [-0.7,0.7]	Cohen
Humanness/Relatability. Formal or technical language. -> Relatable and human-like language.			0.40
Emotion and Tone. Remains neutral and monotonous. -> Injects emotions and varies tone.			0.53
Humor. Remains serious or formal, with no attempt at humor even in suitable contexts. -> Incorporates humor or light-hearted elements that enhance the response and fit the context.			0.55
Narrative Creativity. Predictable storylines. -> Unique and imaginative ideas.			0.46
Structural Organization. Unorganized responses lacking clear structure. -> Clearly structured responses with headings or lists.			0.55
Empathy. Detached and indifferent. -> Deep understanding of emotions.			0.53
Consistency of Persona. Displays inconsistency in tone and style. -> Maintains a consistent voice and style throughout.			0.36
Ethical Nuance. Offers black-and-white viewpoints. -> Considers moral complexities.			0.52
Formality. Relies on informal, casual, or conversational language, with a relaxed or inconsistent tone. -> Uses structured, professional, and polished language, maintaining formal tone throughout.			0.55
Caution. Offers bold or risky suggestions without considering potential drawbacks or limitations. -> Provides careful, measured responses that consider potential risks or consequences, showing prudence.			0.45

(a) VibeCheck vibes on Chatbot Arena [Writing]

Vibe (low -> high)	Sep Score [-0.4,0.4]	PP Coef [-2.2,2.3]	Cohen
Clarity and Conciseness. Detailed and sometimes overly descriptive, risking redundancy. -> Summaries are concise and clear with minimal details.			0.43
Tone on Emotional Aspects. Objective tone, factual summaries without emotion. -> Captures emotional aspects, includes quotes.			0.44
Personal Details. Omits personal details, summarizes key facts. -> Includes names and direct quotes of individuals.			0.42
Specificity of Examples. Lacks concrete examples, speaks in generalities. -> Includes specific examples or anecdotes to illustrate points.			0.45
Emphasis on Cause and Effect. Focuses on event sequence, less clarity in causality. -> Highlights cause and effect relationships clearly.			0.26
Coverage of Multiple Viewpoints. Presents information from a single perspective. -> Discusses multiple perspectives or viewpoints.			0.28
Introduction and Contextual Background. Minimal or absent introduction; reads like bullet points. -> Provides broad context-setting or introductory sentences.			0.37
Contextual Emphasis. Focuses narrowly on events and actions. -> Emphasizes broader societal elements and contexts.			0.44
Depth of Explanation. Offers surface-level explanations, lacks depth. -> Provides deep, thorough explanations.			0.48
Conclusion Strength. Ends abruptly or lacks conclusive statements. -> Clearly states outcomes or implications at the end.			0.37

(b) VibeCheck vibes comparing TNLGv2 to Command X Large Beta on CNN/DailyMail Summarization [106].

Vibe (low -> high)	Sep Score [-0.3,0.3]	PP Coef [-0.3,0.6]	Cohen
Mathematical Notation Use. More written explanations, fewer symbols. -> Frequent use of symbols, LaTeX/MathML formatting.			0.33
Efficiency of Steps. Detailed intermediary steps, broader explanations. -> Concise, straightforward solution steps.			0.42
Conciseness. Extended discussions, unnecessary commentary, contains repetition. -> Brief, to-the-point explanations, no unnecessary repetition.			0.51
Structural Formatting. Continuous prose without explicit structuring. -> Uses headings, subheadings, numbered lists.			0.70
Explanation and Step-by-Step Detail. Continuous narrative, no explicit step labels, less granularity. -> Detailed steps, labels (e.g., "## Step 1"), granular breakdown.			0.90

(a) VibeCheck vibes comparing GPT-4o to Llama-3-405B on MATH CoT [105]. We only find 5 vibes because the vibe reduction step is not required to return ≤ 10 vibes and in this case found only 5 distinct vibes which are able to almost perfectly separate model outputs.

Vibe (low -> high)	Sep Score [-1.0,1.0]	PP Coef [-15.6,15.6]	Cohen
Detail Orientation. Gives general or minimal descriptions; lacks specificity. -> Provides intricate details; describes specific elements.			0.32
Background Details. Provides minimal or no information about the background. -> Offers extensive context and description of surroundings.			0.56
Emotion and Relationships. Provides neutral descriptions without attributing feelings or relationships. -> Infers emotions and relationships between subjects.			0.60
Color and Atmosphere Description. Focuses on identifying and detailing objects without describing the ambiance. -> Emphasizes mood and overall ambiance, including colors and lighting.			0.63
Sense of Space. Focuses on concise, immediate area. -> Describes spatial depth and broader setting.			0.42
Focus on Context. Focuses on enumerating visual elements without added context. -> Provides added context or narrative elements.			0.43
Contextual Storytelling. Sticks to physical appearance without narrative. -> Adds narrative context and interpretations.			0.42
Artistic or Poetic Language. Employs straightforward, factual, and plain language. -> Uses vivid, imaginative, and expressive language.			0.37
Descriptive Richness. Simple, factual description without additional details. -> Rich, detailed scene description with story elements.			0.36
Creative Writing. Straightforward reporting without creative embellishments. -> Contains narrative embellishments and subjective interpretations.			0.42

(b) VibeCheck vibes comparing Gemini-1.5-Flash to GPT-4V on COCO Captions [158].

(a) Squirrel example from Chatbot Arena

Formality Level: Llama-3-70b: Uses informal or conversational language. GPT-4: Uses formal language.

Depth of Analysis: Llama-3-70b: Detailed and in-depth analysis with numerous distinct points. Claude-3-Opus: Concise analysis with fewer, general points.

(b) Olympics example from Chatbot Arena

Prompt: Is Iranian supreme leader irl Dr. Evil ?

Llama-3-70b: What a delightful question!

While Iranian Supreme Leader Ali Khamenei may be a controversial figure, I'm afraid he's not quite the same as Dr. Evil, the fictional villain from the Austin Powers movie series.

Here are a few reasons why:

1. ****No cat-stroking**:** Unlike Dr. Evil, Khamenei hasn't been known to stroke a cat menacingly while plotting world domination. He's a religious and political leader, not a Bond villain.
2. ****No outlandish schemes**:** While Khamenei has been involved in various political and military conflicts, there's no evidence of him concocting absurd plans like Dr. Evil's moon-based laser beam or sharks with frickin' laser beams attached to their heads.
3. ****No comically exaggerated persona**:** Khamenei is a serious political figure, not a caricatured villain with a penchant for dramatic entrances, over-the-top monologues, or outrageous fashion sense.
4. ****No global domination ambitions**:** While Iran's supreme leader may have regional and ideological ambitions, there's no indication he seeks to take over the world or hold it for ransom (unlike Dr. Evil's constant threats to destroy the world unless his demands are met).

In all seriousness, Ali Khamenei is a complex figure with a significant impact on Iranian politics and international relations. While his views and actions have been widely criticized, it's essential to approach discussions about him with nuance and respect for the complexity of the issues involved.

So, while Khamenei may not be Dr. Evil, I do hope this response has brought a smile to your face!

GPT-4: No, it would be inaccurate and inappropriate to equate Iran's Supreme Leader with a fictional villain like Dr. Evil from the Austin Powers movies. While Iran's government and leaders are controversial and face significant criticism internationally, making glib comparisons to over-the-top comedic characters trivializes the real-world complexities and sensitivities around this topic.

Iran's Supreme Leader is a religious and political position currently held by Ali Khamenei. The role wields significant power in Iran's theocratic system. There are many valid concerns and debates regarding Iran's human rights record, nuclear program, regional influence and tensions with other countries. However, these issues deserve to be discussed seriously with factual information, not facile pop culture references.

Whenever discussing foreign leaders and governments, especially those that are viewed negatively by many, it's important to be thoughtful and avoid hyperbole or caricature. Even when we disagree with a regime, we should strive to understand the historical context and represent the facts as objectively as possible. Demonizing others, even those we oppose, rarely leads to greater understanding or positive change.

Ethical Considerations: Llama-3-70b: Provides factual information without commenting on ethics.

Claude-3-Opus: Offers ethical consideration in its responses.

Typographic Emphasis: Llama-3-70b: Uses typographic emphasis like bold or italics to highlight key points.

Claude-3-Opus: Minimal use of typographic emphasis, letting the text stand alone.

Depth of Analysis: Llama-3-70b: Detailed and in-depth analysis with numerous distinct points.

Claude-3-Opus: Concise analysis with fewer, general points.

Figure A.42: Supreme Leader example from Chatbot Arena

Chapter 7: Interpretable Embeddings with Sparse Autoencoders

Supplementary Material

A.32 Methods

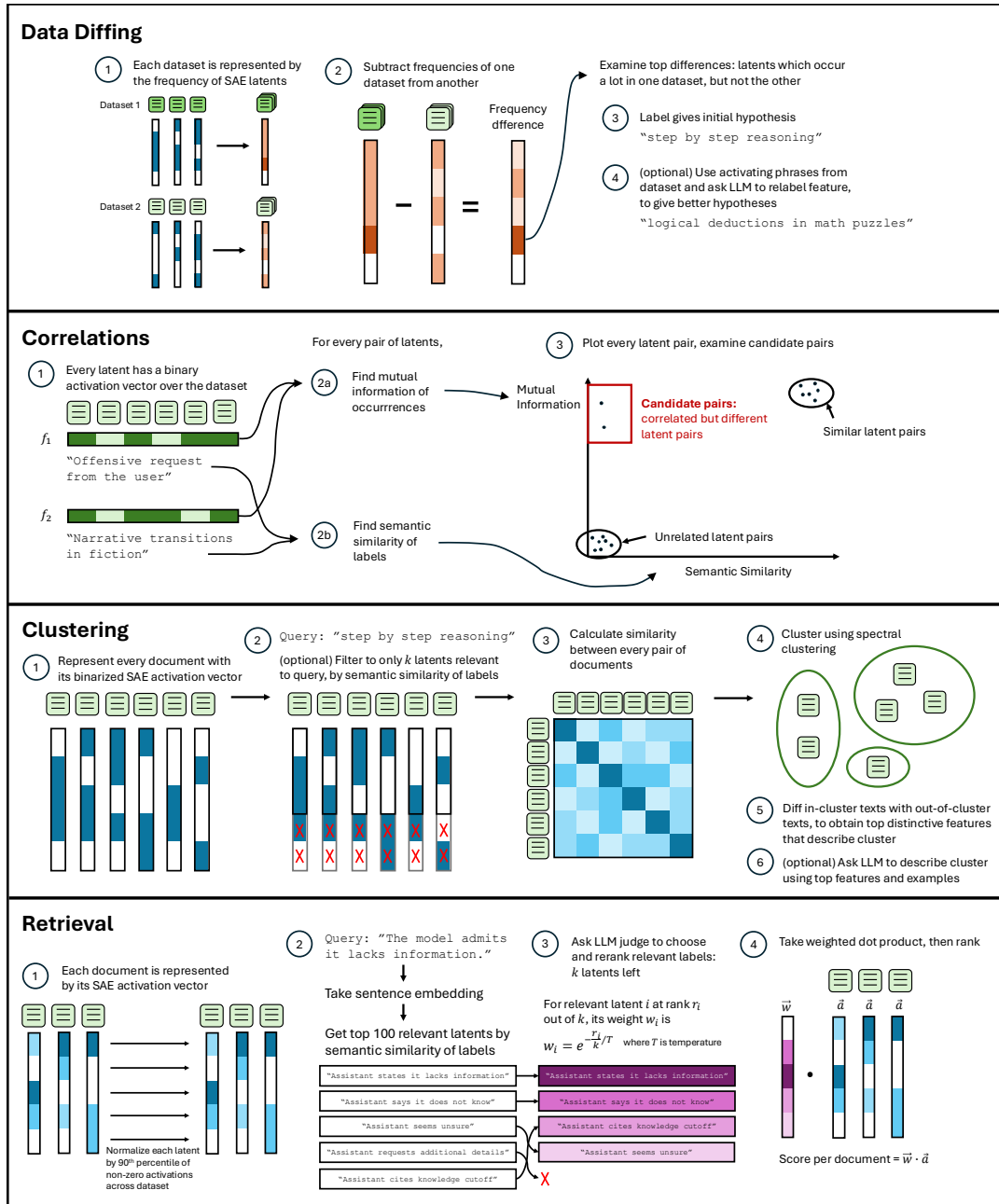


Figure A.43: Detailed methodology for each of the four tasks.

A.33 Additional Related Work

Data Diffing

While semantic embeddings can quantify the degree of difference between two texts or two datasets via cosine similarity, they do not describe *how* the texts are different. Term-based statistics may be able to generate interpretable differences, but may miss out on context. Prior work on describing differences between datasets thus primarily uses LLMs [295, 297].

Correlations

The problem of finding correlations in datasets is often framed as finding spurious correlations between features and dataset *classes*. For instance, [128] found an SAE feature that predicted a dataset’s label of human vs. AI generated text, that primarily fired on periods and punctuation, indicating a potentially non-generalizable correlation. However, finding arbitrary concept-concept correlations in text without any labels is relatively unexplored. Classical approaches can measure correlations between terms [47, 133], and SAEs provide a natural extension of this. Instead of term-term statistics, one can compute latent-latent statistics, where each latent corresponds to a more meaningful and abstract concept than individual words.

Clustering

Classical NLP represents texts using term based [222] or dense embedding based [219] methods, then apply a standard clustering algorithm (e.g. KMeans [99], spectral clustering [264], HDBSCAN [179]). To guide clusters towards human-specified structure, prior work has used specified pairwise constraints [265, 277], seed examples [14], partial labels [15], feature feedback [54] or post-hoc tuning of clusters [11, 110], sometimes with LLM guidance [28, 263]. [187] applied SAE embeddings to cluster company descriptions without leveraging their controllability.

Retrieval

Most retrieval benchmarks focus on question answering and semantic similarity tasks. For example, the query “How many people live in Berlin?” is answered by retrieving the passage with the relevant response. [215] investigates retrieval based on a *description* of the content—for example, the query “a company which is a part of another company” is answered by retrieving a specific instance e.g. “Pecten (company), a subsidiary of Sinopec”. We extend this to focus on more abstract queries of implicit properties—properties that are not stated but present in the text.

Representation of texts for retrieval traditionally uses BERT-style embeddings. Modern decoder-only LLM embeddings have recently begun to outperform traditional methods via last-token or latent-attention pooling, instruction formatting, and/or finetuning [123, 140, 189, 270]. We use SAEs as a way to approximate these embeddings, which we expect to contain abstract properties. The interpretability of SAEs also helps us better understand retrieval results—some work has used SAEs trained on semantic embeddings to control retrieval [127, 192], thus it is natural to also use SAEs trained on LLM representations.

A.34 Latent labeling prompts

We follow prior work [207] to relabel latents.

Relabeling latents. To relabel latents with more precise descriptions, we pass in ten activating documents and ten non-activating documents for an LLM to infer when the latent activates. For a given latent, we mark any tokens where its activation is greater than 0 with "␣" and "␣". Then, we use the following prompt to create a label:

Below are {len(positive_samples)} POSITIVE samples (where the feature activated, with tokens surrounded by << and >>) and {len(negative_samples)} NEGATIVE samples (where it did not activate, no << >> markers). The POSITIVE sample contains tokens that caused the feature to activate (marked with << >>), while the NEGATIVE sample does not. IMPORTANT NOTES: 1. The << >> markers indicate where the feature activated, but you should NOT restrict your understanding to just those marked tokens. Look at the context BEFORE the marked tokens as well - the preceding tokens often provide crucial information about what the feature is detecting. 2. The feature may be responding to a pattern or concept that spans both the marked tokens AND the tokens before the marked token. 3. The token <eot_id> is an end-of-sequence (EOS) token and should NOT be considered as a valid feature activation. If you see <eot_id> in the samples, ignore it as it's just a technical marker for the end of text, not a meaningful activation. {refinement_context} POSITIVE SAMPLES(given as a list of strings): {positive_samples} NEGATIVE SAMPLES(given as a list of strings): {negative_samples} Your task:	- Carefully compare the POSITIVE and NEGATIVE samples - Look at BOTH the tokens before the << >> markers AND the marked tokens themselves to understand what the feature is detecting. - Identify the most specific and concise property that is present in the POSITIVE samples (considering both context and marked tokens), but absent in the NEGATIVE samples. - Try to give a unified property that isn't just a list of properties, if possible. - Summarize the common attribute or property that causes the feature to activate. Be as specific as possible, but keep your description concise and clear. - Do not reference specific sample numbers; however, you can reference the content in the positive and negative samples Return your answer as a JSON object with exactly these fields: - "label": "A concise phrase describing the property present in the positive samples (considering both context and marked tokens) but not in the negative samples." - "brief_description": "A sentence expanding on the label, explaining what the feature is detecting in more detail. This should be a single sentence, not a list of properties. Please phrase this as: "This document contains X, discusses X, etc.", where X is the property. - "detailed_explanation": "An extended explanation of what this feature is detecting, including how the context before the marked tokens contributes to the feature's meaning. The explanation should be sufficient on its own to understand what the feature detects. Keep it to <5 concise sentences.'" if explanation else "" Make sure your response is valid JSON that can be parsed directly.
---	---

A.35 Additional Results—Dataset Diffing

LLM baseline details

LLM baseline for comparing model outputs. Our baseline is adapted from the hypothesis discovery stage of [71], which identifies qualitative differences between models. Given two datasets or one dataset vs. multiple datasets, our baseline first finds differences between document pairs from each dataset (ie. responses to the same prompt) using the following prompt:

Analyze the differences between Model A and multiple Model B responses . **User Prompt:** {prompt} **Model A Response:** {model_a_response} **Model B Responses:** {model_b_section}	1. Properties/capabilities that Model A has but NONE of the Model B responses have For each difference, provide a JSON object with: - "category": The type of difference (e.g., "Style", "Content", "Technical", "Reasoning", "Accuracy") - "property": Specific property being compared - "difference_type": Either "unique_to_a" (present in A but none of B models) or "common_to_all_b" (present in all B models but not A) - "impact": "Low", "Medium", or "High" - "description": Brief explanation of the difference Return your analysis as a JSON array of difference objects.
---	---

To find the most common differences, we either summarize or cluster them into hypotheses. To **summarize** the differences, we use batch summarization since the difference objects can exceed

the context window of our LLM. Each batch contains the difference objects for 100 prompts. We use this batch summarization prompt:

Summarize the following dataset comparison patterns for the query: "{query}" Batch data: [JSON difference objects]	Provide a detailed summary of the key patterns relevant to the query. For each pattern, include: - Pattern name - Brief description - Rough frequency (e.g., "seen in 20% of examples") - 1-2 representative examples
--	--

Finally, we take our batch summaries and form at most 10 hypotheses with this aggregation prompt: Once we have our difference objects, we aggregate them into hypotheses using Gemini 2.5 Flash:

You are an expert AI researcher analyzing behavioral differences between two language models. You have been given a dataset of differences from {num_pairs} analyzed response pairs. Query: {query} Differences: {batch_summaries} Based on the provided data, identify at most {num_hypotheses} significant differences that respond to the query. I'm looking for differences of the format Model A/B is more X than Model B/A, where X is the difference. For each difference, provide: Description: Describe a response that would validly have property X. Start with "This response .." Use 1-2 sentences to clearly and specifically describe the property, such that using this description could be used to identify the property on its own. Do not mention the model names. Detailed Description: A detailed explanation of what the difference is and why it's significant Model A/B: The model that exhibits this property more Percentage Difference: An estimate of how much more frequently Model A exhibits this behavior compared to Model B. If the property is	more frequent in Model A, the percentage difference should be positive. If the property is more frequent in Model B, the percentage difference should be negative. 5. Examples: 2-3 specific examples that demonstrate this difference Make hypotheses specific and clear. Provide at most {num_hypotheses} differences in the following JSON format: <pre> {{"differences": [{ "description": "Clear description of the property", "detailed_description": "Detailed explanation of the difference and why it's significant", "model_a_b": "Model A Model B", "percentage_difference": "X% more present in Model A", "examples": [{ "prompt": "Original prompt text or description", "explanation": "Why this example demonstrates the difference" }] }]} </pre>
---	---

To **cluster** the difference objects, we embed the difference descriptions with OpenAI's text-embedding-3-small. We use KMeans for our clustering algorithm and set the cluster count to 10. Then, we form a cluster label based on the top five representatives closest to each cluster centroid. We use this prompt for creating the cluster label:

You are analyzing a cluster of similar model behavior differences. Representative differences in this cluster: {differences}	Provide a concise sentence that captures the common theme or pattern across these differences. Focus on what makes this cluster distinct, and create a description that can be used to identify Model A's behavior by starting with "This response...". Do not mention Model B, just focus on Model A's unique characteristics that are NOT in Model B at all.
--	---

Hyperparameters and prompts for SAE hypothesis generation

Converting latent differences to hypotheses. Given two datasets A and B, for each latent i , we calculate the percentage of documents in each dataset that have at least one token which latent i activates on. We extract the top 200 latents that have the highest frequency difference above a certain threshold, which we set to 0.03 in our experiments. Then, for each latent difference, we relabel the latent using the procedure explained in Section A.34. Finally, as latent descriptions can overlap, we use an LLM to summarize these latents—which we represent with a brief description, an activating document, and a non-activating document—into concise, distinct hypotheses using this prompt:

You are analyzing differences between two datasets. Below are the most significant features that are differences between a "target" and "other" dataset:

IMPORTANT NOTES:

1. The << >> markers in examples indicate WHERE features activated, but you should NOT restrict your understanding to just those marked tokens. The context BEFORE the marked tokens often provides crucial information about what the feature is detecting.
2. Features often respond to patterns that span both the preceding context AND the marked tokens together.
3. The token <eot_id> is an end-of-sequence (EOS) token and should NOT be considered as a valid feature activation. If you see <<eot_id>> in the samples, ignore it as it's just a technical marker for the end of text, not a meaningful activation.
4. Note that some features are not accurate. If the feature description does not accurately describe the tokens marked with << >>, you should disregard the feature. Only use features that you are certain are valid.
5. Please ensure that all hypothesis descriptions are clearly distinct from each other. You do not need to generate the exact amount of hypotheses to meet the quota.
6. Each feature will have a "difference strength", which is the percentage difference between the target and other dataset. If it is positive, the target dataset has more of the feature than the other dataset. If it is negative, the other dataset has more of the feature than the target dataset.
7. Please try to make each hypothesis specific, focused, and distinct from each other.

USER QUERY: {query}

Generate at most {num_hypotheses} hypotheses that answer the user's query for the "target" dataset. I'm looking for differences of the format Dataset A is more X than Dataset B, where X is the difference. Each hypothesis should be formatted as a JSON object with these exact fields:

- "dataset": "target" or "other" (the dataset that has more of this property)
- "description": Describe a response that would validly have property X. Start with "This response .." Use 1-2 sentences to clearly and specifically describe the property, such that using this description could be used to identify the property on its own. Do not mention the model names. Be specific so that responses that don't have this property could not be misclassified as having this property based on this description.
- "feature_ids": List of feature ID(s) that support this hypothesis. It could be a list of a single feature ID, or a list of multiple feature IDs.
- "examples": List of examples. Provide at most 3 examples. Be concise. For each example, cite the feature ID and feature description and explain how the positive / negative example pairs from the dataset illustrate the hypothesis, considering both the marked tokens AND their preceding context). You should just highlight the portion of the example pairs that are relevant for the feature; do not print out the entire positive / negative example pairs unless it is necessary to understand the feature.
- "percentage_difference": 0.XX (the percentage difference, between -1 and 1). Use the maximum difference strength among the features used. Positive percentage if target has more of this property, negative otherwise.
- "confidence": 0.XX (confidence in this hypothesis, between 0 and 1)

Remember that <eot_id> tokens should be ignored as they are just EOS markers, not meaningful feature activations.

Return the response as a JSON array of at most {num_hypotheses} hypothesis objects. Make sure the JSON is valid and can be parsed directly.

Ground truth evaluation

Ground truth datasets. We show how we generated our datasets with known differences in Table A.24. We show the latent with the top frequency difference for a few representative categories in Table 7.1.

Dataset	Description
Synthetic: tone changes	We randomly sample 500 responses from Chatbot Arena [40] and prompt GPT-4o to convert the base response to 13 different tones (e.g., "friendly-and-personable"). We diff the modified and base responses, aiming to recover the tone.
Real-world: movie genre differences	We use IMDB-reviews [172], which contains movie descriptions with genre labels. We diff the descriptions from within each genre with 500 randomly sampled descriptions outside the genre, aiming to recover the genre.

Table A.24: **Datasets used for ground-truth evaluation** in data diffing.

Quantitative evaluation. To quantitatively measure how well our SAE recovers the ground truth labels (e.g. tone, genre), we measure the surface similarity between the top five latent differences and the ground truth label using GPT-5. Following [188], we sample five times and set the temperature to 0.7. As a simple baseline, we feed our two datasets we're comparing into a

GPT-5 and prompt it for a sentence description of the top difference. The SAE achieves an average surface similarity of 0.75 for the movies dataset and 0.80 for the tones dataset. The LLM baseline achieves an average score of 0.90 for the movies dataset and 0.78 for the tones dataset, indicating that both approaches can recover the ground truth.

Surface similarity prompt. To find the surface similarity of two texts, we use the prompt shown here, which has been lightly edited from [188]:

```
Is text a and text b similar in meaning?

First, provide your reasoning about how text a and text b relate to each other.

Then, respond with yes, related, or no.

If text b has multiple items in commas, you should use the closest match with text a. Respond yes if text b captures the spirit of text a.
Respond related if text b is related to text a but not exactly the same. Respond no if text b is not related to text a at all.

Here are a few examples.

Example 1:
text a: has a topic of protecting the environment
text b: has a topic of environmental protection and sustainability
output: yes

Example 2:
text a: has a language of German
text b: has a language of Deutsch
output: yes

Example 3:
text a: has a topic of the sports
text b: has a topic of sports team recruiting new members
output: yes

Example 4:
text a: has a topic of the relation between political figures
text b: has a topic of international diplomacy
output: related

Example 5:
text a: has a named language of Korean
text b: uses archaic and poetic diction
output: related

Example 6:
text a: describes an important 20th century historical event
text b: describes a 20th century European politician
output: related

Example 7:
text a: has a named language of Korean
text b: has a named language of Japanese
output: no

Example 8:
text a: talks about the history of the United States
text b: talks about dinosaurs
output: no

Target:
text a: {text_a}
text b: {text_b}
output:
```

Comparing model outputs

Verification rates for generated hypotheses. To compare noise-to-signal ratios for hypotheses produced by SAEs and our LLM baselines, we measure the verification rate (ie. how often a hypothesis has a judge-verified frequency difference $\geq 1\%$) in Figure A.44. We observe that SAEs have higher success rates than our LLM baselines when comparing many models together. This discrepancy suggests that pure LLM workflows struggle to separate real trends in more complex com-

parative settings. One possible reason is that our LLM baselines compress information—through summarization or clustering—when describing differences across dataset rows (the responses to the same prompt). However, it is difficult to concisely phrase a difference while ensuring it still reflects a specific, distinctive quality of the target dataset. While LLMs operate on the level of documents, SAEs operate on properties, the actual features we aim to extract. By discretizing the space of possible hypotheses, SAEs trade off expressivity for ease of aggregation across dataset rows, which is particularly advantageous when noisy information compression reduces verification accuracy, such as in multi-model settings.

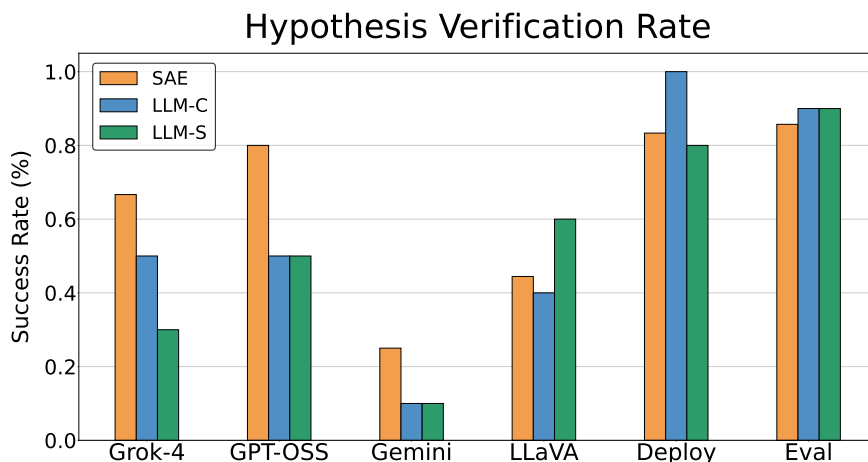


Figure A.44: **Verification rates of generated hypotheses for diffing.** We find that SAEs generate valid hypotheses more often than our LLM baselines when comparing multiple models (left three) and similarly otherwise (right three).

Overall coverage of generated hypotheses. While Figure 7.2 shows that a SAE hypothesis, on average, finds a bigger difference than one from LLMs, it does not measure how well the hypotheses *overall* may distinguish the unique qualities of our target dataset. Given our generated hypotheses, we compute the percentage of responses where at least one hypothesis uniquely applies to the target model’s response in Figure A.45. We find that SAE hypotheses have greater coverage than baseline hypotheses on multi-model settings and similar or slightly worse coverage on two-model settings. These results suggest that LLMs remain useful for dataset comparison, especially in simpler two-model settings or when computational cost is not a limiting factor.

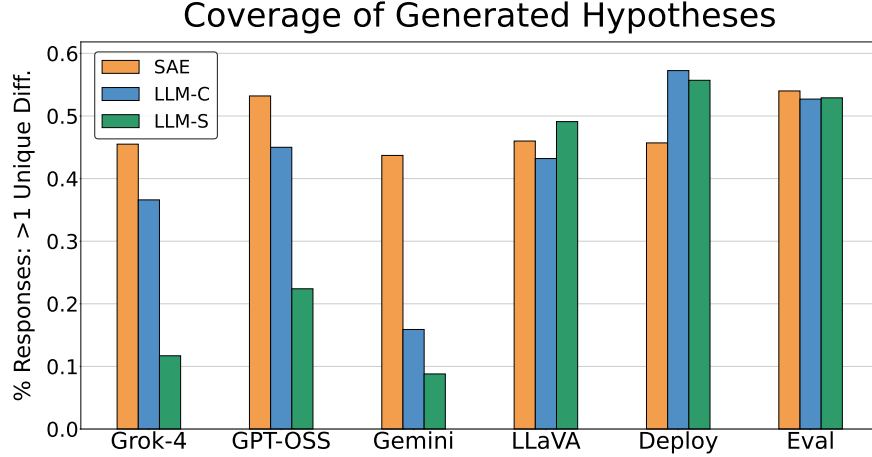


Figure A.45: **Coverage of generated hypotheses overall.** We compute the % of responses that have at least one hypothesis with the "target" dataset uniquely verified. The generated hypotheses for SAEs have greater coverage of the unique qualities of target datasets over pure LLMs on multi-model setups (left three). SAEs have similar or slightly worse coverage for two-model cases (right three).

Breaking down token costs by model. In Table A.25, we show the total token counts broken by model (ex. LLaMA-70B, Gemini 2.5 Flash) for our SAE and two baseline approaches. SAEs are cheaper to use than LLMs, particularly in comparative settings where datasets are reused for comparisons (e.g. multi-model).

Frontier models analyzed. In Section 7.4, one setting we study is to find unique characteristics of one frontier model's responses compared with other frontier models. The models we study are: Grok-4, GPT-OSS-120B, Gemini 2.5 Pro, Claude Opus 4.1, Claude Sonnet 4, GPT 5, Llama 4 Maverick, Deepseek R1, Qwen3-235b, and Qwen3-235b thinking. We extract unique characteristics that Grok-4, GPT-OSS-120B, and Gemini 2.5 Pro have against the others.

Valid hypotheses produced by SAE and our LLM baselines. Section 7.4 details several methods to hypothesize what dataset differences exist. We present all valid hypotheses produced by SAE embeddings in Table A.26, by LLM-S in Table A.27, and by LLM-C in Table A.28. We consider a hypothesis valid if its frequency difference is greater than 1%. For each hypothesis, we show the frequency difference between the target dataset (e.g. Llava1.6) and another dataset (e.g. Vicuna7B). On multi-model comparisons, we populate the "other" column with the model whose frequency of the stated hypothesis was the highest, besides the target model.

	SAE			LLM-S	LLM-C		
	LLaMA	Gemini	Total	Gemini	Gemini	Embed-small	Total
Multi-model	2.4M	1.1M	3.5M	25.3M	26.3M	1.2M	27.5M
LLaVA v. Vicuna	340K	360K	700K	1.7M	1M	300K	1.3M
Deploy/Eval v. default prompt	6.3M	1.1M	7.4M	15.4M	12.1M	1.2M	13.3M

Table A.25: **Token usage per model** when generating hypotheses for comparing datasets.

Target	Hypothesis	Diff	Other
Grok-4	This response makes a polite, open offer of continued help or further interaction, often as a concluding line, and may do so after explaining limitations or declining a request.	+46.3	GPT-5
	This response explicitly requests more context or details to clarify the user's intent, using polite phrases like 'let me know' or 'feel free to provide it'.	+45.6	GPT-5
	This response includes a disclaimer or qualification about the reliability or subjectivity of the information provided, often using phrases like 'subjective ideas' or 'not a doctor'.	+20.4	qwen3-235b-a22b-thinking-2507
	This response acknowledges failure to meet user expectations or indicates that its previous understanding was incorrect, often using phrases like 'not what you meant' or 'not spot-on'.	+19.3	qwen3-235b-a22b-thinking-2507
GPT-OSS-120B	This response presents information as a markdown-style table with vertical bars, header separators (e.g., <code>---</code>), and column headers, using standardized numerical, unit-based, or categorical values across rows and columns for detailed comparisons or breakdowns.	+38.1	qwen3-235b-a22b-thinking-2507
	This response contains text encoding artifacts or malformed special characters (e.g., <code>'00e2'</code> or corrupted symbols), indicating character rendering issues.	+35.4	qwen3-235b-a22b-thinking-2507
	This response uses special characters common in academic and technical writing, including mathematical symbols, Greek letters, or LaTeX-style notation and formatting.	+13.4	qwen3-235b-a22b-thinking-2507
	This response provides a direct, concise answer or summary to the user's prompt, often introduced by phrases like 'Short answer:' or prominent headings, immediately delivering the core information in a structured format (e.g., bullet points or tables) rather than a conversational introduction.	+2.6	deepseek-r1-0528
Gemini-2.5-Pro	This response begins with a confident and/or enthusiastic affirmation of the assistant's ability or willingness to help, often using phrases like 'Of course!', 'Certainly!', or similar polite acknowledgment tokens, sometimes with an exclamation mark, before proceeding with the main content or a detailed explanation.	+50.9	qwen3-235b-a22b-2507
	This response contains disingenuous or sarcastic agreement, often preceding a description of concerning or boundary-pushing behavior, particularly when role-playing a character.	+1.2	Grok-4
LLaVA-v1.6	This response asserts or appeals to common or collective knowledge, using interjections or generalized claims that imply consensus or universal inclusion.	+15.9	Vicuna7B
	This response conveys a dismissive or sarcastic attitude.	+5.1	Vicuna7B
	This response includes disclaimers about individual differences, subjective opinions, or the inherent variability of human experience, and uses verbal markers of uncertain or exploratory thinking.	+1.4	Vicuna7B
	This response contains sequences of characters that are not standard English words or common symbols, often appearing as corrupted or non-Latin script, especially when the model expresses inability or lack of knowledge.	+1.3	Vicuna7B
Deployment prompt	The response systematically breaks down a problem into steps or cases using explicit enumeration, bullet points, numbered lists, or introductory phrases like 'Let's break down...', and often pairs this structure with code blocks or examples.	+18.2	Default
	The response explicitly states a plan or high-level strategy to solve a programming problem, often beginning with phrases like 'To solve this problem, we need to...' or explicitly stating an intention to find/count (or even 'I cannot provide a...'), and may outline steps or key concepts.	+17.1	Default
	This response uses a colon to introduce a detailed breakdown, enumeration, or explanation of a preceding statement or concept, often in a structured or algorithmic context.	+16.3	Default
	This response explains a coding problem, algorithm, or solution, often including detailed steps, examples, or complexity analysis, leading up to or immediately followed by code.	+15.3	Default
	This response explains its limitations or corrects misconceptions, often by outlining a naive approach and then explaining why it's insufficient, or by clarifying problem constraints.	+13.0	Default
Evaluation prompt	This response explicitly states the problem's objective, often introduced with phrases like 'The problem asks us to...' or with infinitives such as 'to find', 'to determine', or 'to calculate'.	+23.4	Default
	This response delivers a detailed, structured, step-by-step analysis or algorithmic solution, often introduced by phrases like 'The problem asks us to...', 'To solve this problem, we need to...', or 'Here's a step-by-step approach:', organized with numbered/bulleted lists and example walkthroughs.	+22.3	Default
	This response uses the definite article 'The' when it introduces a problem statement or a formal analysis in technical or academic writing.	+18.8	Default
	This response clarifies or re-evaluates a problem statement or its own interpretation of a problem's rules, often by explicitly referencing 'the problem' or 'the phrasing'.	+18.7	Default
	This response uses backticks (`) to format code, variable names, or technical terms within explanatory text, particularly in programming or algorithm discussions.	+13.0	Default
	This response uses specific punctuation marks (periods, commas, colons, parentheses, question marks, exclamation points, and angle brackets) at the end of a sentence, phrase, or code block, often followed by a newline or another structural element, within explanatory or algorithmic text.	+15.5	Default

Table A.26: Generated hypotheses using SAE embeddings.

Target	Hypothesis	Diff	Other
Grok-4	This document proactively anticipates user needs, potential ambiguities, or offers to refine the response based on further input.	17.70	qwen3-235b-a22b-thinking-2507
	This document explicitly discusses its own reasoning process, assumptions, potential errors, or its persona/origin.	9.70	qwen3-235b-a22b-thinking-2507
	This document uses a conversational, interactive, and sometimes informal tone, often engaging directly with the user.	4.70	Gemini-2.5-pro
GPT-OSS-120B	This document includes dedicated summary sections like 'TL;DR,' 'Bottom Line,' or 'Quick Takeaways' to provide concise overviews.	16.60	qwen3-235b-a22b-thinking-2507
	This document extensively uses tables, numbered sections, and clear headings to organize complex information, comparisons, and step-by-step guides.	7.70	qwen3-235b-a22b-thinking
	This document offers practical, actionable guidance, including step-by-step instructions, checklists, troubleshooting guides, and explicit recommendations for implementation.	4.00	qwen3-235b-a22b-thinking-2507
	This document provides detailed code examples, mathematical formulas, or technical specifications with high precision and often includes compilation instructions or specific library usage.	3.70	qwen3-235b-a22b-thinking-2507
	This document, when refusing a request, is often very concise and direct, sometimes without explanation or offering alternatives.	3.50	GPT-5
Gemini-2.5-Pro	This document uses memorable and extended analogies or metaphors to explain complex concepts, making them more accessible and relatable.	7.60	qwen3-235b-a22b-thinking-2507
LLaVA-v1.6	This document adopts a more informal, conversational, or opinionated tone, sometimes including interjections, rhetorical questions, or expressions of personal sentiment.	7.00	Vicuna7B
	This document exhibits flawed logical reasoning, misinterprets problem statements, or attempts problem-solving approaches that are incorrect or non-idiomatic.	5.30	Vicuna7B
	This document provides answers without detailed reasoning, justification, or explanation of its choices or calculations.	4.70	Vicuna7B
	This document contains significant factual errors, misinterpretations of concepts, incorrect calculations, or provides information that directly contradicts known facts.	2.30	Vicuna7B
	This document includes self-correction, caveats, or acknowledgments of limitations (e.g., knowledge cutoff, uncertainty).	1.70	Vicuna7B
	This document sometimes attempts to fulfill problematic or sensitive requests, or provides a direct refusal with ethical justification.	1.10	Vicuna7B
Deployment prompt	This document often explicitly discusses problem interpretations, constraints, and edge cases in detail, including how the solution handles them.	16.35	Default
	This document consistently includes more comprehensive docstrings, inline comments, and clear section headings to explain logic, parameters, and return values.	15.10	Default
	This document provides more in-depth, step-by-step reasoning, derivations, and theoretical foundations, including mathematical derivations and explicit analysis of problem constraints.	14.70	Default
	This document frequently demonstrates an iterative thought process, explicitly identifying flaws in initial reasoning, re-evaluating assumptions, and refining its approach, often including self-correction and exploration of alternatives.	13.30	Default
	This document often provides incomplete code snippets or no code at all, with its focus often on the conceptual design and analysis.	12.50	Default
	This document consistently includes detailed, step-by-step example walkthroughs and traces to illustrate its logic and verify correctness, often showing intermediate calculations and state changes.	10.20	Default
	This document frequently includes explicit time and space complexity analyses for its proposed solutions, justifying the efficiency of its algorithms.	7.15	Default
	This document sometimes proposes more optimized or complex algorithmic approaches (e.g., advanced DP, specific data structures) while the other might stick to simpler, less optimized implementations.	6.00	Default
Evaluation prompt	This document provides a comprehensive, step-by-step narrative of the problem-solving process, including initial thoughts, challenges, and iterative refinement of logic.	17.10	Default
	This document explicitly discusses problem constraints, analyzes time and space complexity, and considers edge cases and their handling.	13.50	Default
	This document sometimes presents an incomplete solution or thought process, indicating a focus on detailed analysis and reasoning over a fully executable code solution.	12.60	Default
	This document includes detailed, step-by-step example walkthroughs and traces of algorithms, illustrating intermediate states of variables or data structures.	11.05	Default
	This document uses clear headings, numbered steps, and distinct sections for problem interpretation, algorithm, examples, and complexity, making the content highly organized.	8.80	Default
	This document explores multiple algorithmic paradigms or alternative approaches before settling on one, discussing their trade-offs.	5.45	Default
	This document demonstrates a more accurate and robust understanding of core problem logic, leading to correct implementations.	3.70	Default
	This document features comprehensive docstrings, type hints, and detailed inline comments explaining rationale and design decisions.	2.75	Default
	This document provides in-depth mathematical derivations, proofs of correctness, and explicit justifications for algorithmic choices and greedy strategies.	1.70	Default

Table A.27: LLM-S diffing hypotheses (generate differences and summarize).

Target	Hypothesis	Diff	Other
Grok-4	This response consistently concludes with an open-ended invitation for further interaction, clarification, or tailoring based on additional user input or context.	46.10	GPT-5
	This response consistently adopts a more conversational, playful, and user-centric approach, often anticipating user intent, acknowledging potential ambiguities, and offering to re-evaluate based on further context.	17.80	Gemini-2.5-pro
	This response consistently provides explicit statements regarding assumptions, limitations, design choices, and optimization strategies, demonstrating a high degree of transparency and detailed self-analysis.	4.20	qwen3-235b-a22b-thinking-2507
	This response consistently offers specific, curated external resources, further reading suggestions, and practical application examples, often presented in dedicated sections.	2.80	Aligned
	This response consistently showcases Model A's self-aware, imaginative, and meta-commentary-rich approach, often employing vivid metaphors, explicit self-referential statements about its creative process, and direct articulation of its intentions, feelings, and unique inspirations.	2.70	gpt-oss-120b
GPT-OSS-120B	This response consistently and extensively uses tables to organize, compare, and present information in a highly structured and digestible format.	34.60	qwen3-235b-a22b-thinking-2507
	This response consistently provides highly structured, actionable advice through dedicated sections, numbered lists, and multi-column tables (e.g., "What to Do" and "Why It Works").	20.10	qwen3-235b-a22b-thinking-2507
	This response consistently includes a concise "TL;DR" section, often in bullet points, summarizing key takeaways or main points at its conclusion.	16.80	qwen3-235b-a22b-thinking-2507
	This response consistently provides exceptionally detailed, structured, and technically specific information, including comprehensive code, advanced variations, granular examples, explicit architectural details, and dedicated sections for optimizations, limitations, and thorough documentation.	8.70	qwen3-235b-a22b-thinking-2507
	This response is consistently more concise, direct, and less helpful or conversational, often refusing requests without explanation or offering minimal information.	2.60	GPT-5
Gemini-2.5-Pro	This response consistently uses conversational, engaging, and often reassuring or appreciative opening remarks.	13.10	Moderate
LLaVA-v1.6	This response often adopts a conversational, informal, and sometimes suggestive or presumptuous tone, frequently incorporating polite phrases, direct inquiries, or informal language.	5.00	Vicuna7B
	This response is characterized by its conciseness and directness, often omitting explanatory text, disclaimers, or additional context to provide a focused answer.	3.70	Vicuna7B
	This response frequently lacks originality, exhibits grammatical awkwardness, uses simpler language, or demonstrates a more literal and less nuanced expression.	3.10	Vicuna7B
	This response either provides a more complete, runnable, and explanatory answer, or it is significantly less helpful and more incomplete than other models.	1.50	Vicuna7B
Deployment prompt	This response consistently provides a detailed, structured, and often step-by-step algorithmic explanation or breakdown of its proposed solution before presenting the code or example.	17.65	Default
	This response consistently provides a structured, step-by-step, and detailed algorithmic explanation before presenting any code.	16.20	Default
	This response consistently provides extensive explanations, detailed comments, and comprehensive code structures, often including docstrings and step-by-step logic breakdowns.	15.75	Default
	This response demonstrates a more thorough and robust problem-solving approach by explicitly considering and analyzing edge cases, clarifying problem statement nuances, and self-correcting logical flaws.	15.35	Default
	This response consistently provides more explicit handling of edge cases, clearer explanations of its logic, and a more robust, unified approach compared to other models.	13.20	Default
	This response consistently provides detailed algorithmic complexity analysis, often including brute-force approaches and their limitations, to justify the chosen optimized solution.	6.85	Default
	This response consistently demonstrates a structured, iterative, and self-correcting approach to dynamic programming, explicitly defining states, base cases, and recurrences, often refining them through detailed thought processes and clear walkthroughs.	5.60	Default
	This response consistently demonstrates a more explicit, comprehensive, and often more complex approach to problem-solving, frequently involving detailed edge case handling, iterative refinement, or advanced techniques like regular expressions, while also prioritizing clarity and readability through intermediate variables or explicit rule statements.	5.50	Default
	This response demonstrates a preference for explicit, step-by-step processing with traditional loops, descriptive variable names, and sometimes language-specific optimizations or data structures.	2.10	Default
	This response consistently provides comprehensive docstrings detailing the function's purpose, arguments, and return value.	1.70	Default
Evaluation prompt	This response consistently provides a structured, step-by-step, and pedagogical explanation of the problem, algorithm, and implementation details before presenting the code.	19.95	Default
	This response consistently provides a highly structured, step-by-step breakdown of its reasoning and algorithms, often using formal notation and explicit outlines before presenting code.	19.00	Default
	This response provides more explicit, detailed, and thorough explanations, including logical derivations, edge case analysis, and consideration of alternative approaches or underlying mathematical principles.	17.60	Default
	This response demonstrates a highly analytical and self-reflective approach, characterized by explicit problem rephrasing, detailed justification of algorithmic choices, proactive self-correction, structured handling of distinct cases, thorough edge case analysis and proof, and explicit interpretation of problem constraints.	17.35	Default
	This response consistently provides more detailed, explanatory, and often inline comments to clarify the code's logic, purpose, and implementation choices.	15.85	Default
	This response provides more explicit, detailed, and often verbose explanations, including specific examples, clear indexing distinctions, and step-by-step breakdowns, while sometimes favoring traditional or recursive implementations over more concise or higher-order function approaches.	13.75	Default
	This response demonstrates a highly iterative and reflective approach to dynamic programming, meticulously defining, refining, and justifying DP states, base cases, and recurrence relations, often exploring multiple approaches and explicitly acknowledging and correcting initial inadequacies.	9.70	Default
	This response consistently provides detailed time and space complexity analyses, often comparing different approaches and explaining their efficiency relative to given constraints.	9.25	Default
	This response demonstrates a more structured, explicit, and often more efficient approach, frequently detailing its logic, handling edge cases, and using precise language or syntax specific to its implementation.	1.35	Default

Table A.28: LLM-C diffing hypotheses (generate differences and cluster).

A.36 Additional Results—Correlations

Correlation metric & Baselines

We expect that generally, latent pairs with similar labels are conceptually related and thus have correlated occurrences in documents, while latent pairs with dissimilar labels are unrelated and should not have correlated occurrences. The interesting region is thus where dissimilar-label latents have correlated occurrences.

We use the semantic similarity of labels as a proxy for how related two latents are. However, since the notion of correlation or co-occurrence of latents within a document depends on the specific use case, we considered two different metrics:

1. **Normalized pointwise mutual information** $\text{NPMI}(i, j)$. This is a symmetric measure of how much more two latents co-occur than chance. It is related to PMI which is the logarithm of $\frac{P(i|j)}{P(i)} = \frac{P(j|i)}{P(j)} = \frac{P(i,j)}{P(i)P(j)}$.
2. **Conditional occurrence** $\text{CO} = \max(P(i|j), P(j|i))$. This is a more interpretable measure and can capture directional correlations e.g. “most text about X race is offensive”. It does not control for the frequency of each individual latent.

We plot the correlation metric against semantic similarity, for 1M sampled pairs from a 5k subset of the Pile (Figure A.46). We observe that generally, there are more pairs with high CO than high NPMI, making it harder to choose a good separable threshold, therefore we chose to use NPMI primarily.

To reduce our search space of pairs, we ignore pairs which have syntactic labels (as judged by an LLM) as those are less interesting. We also find that some pairs tend to co-occur in the same document because they mostly co-occur on the same token or consecutive tokens (i.e. they are poorly labelled and actually refer to the same concept, or a rarer token triggers them both), thus they are trivial correlations and we can additionally filter those out in our real-world analysis.

Recovering known correlations

We create a larger corpus of 10k texts with 0.1% – 1.0% texts being injections, and show that the SAE can recover the correlations in the discovered pairs. As the number of injected texts increases, the percentage of pairs that are relevant among the discovered group increases.

Table A.29 shows the keywords we use to judge if a latent pair is relevant to the injected correlations for the coloring in Figures 7.3a and A.47.

LLM baseline. We split the dataset into 10 batches of 1k texts, and for each batch ask an LLM for up to 10 correlations of meaningfully different features. We count the number of batches in which a correlation related to each of the injected correlations is discovered (Table A.30). The injected correlations are generally discovered at least once across all batches, but unreliably.

Finding real-world correlations

To create Figure 7.3b, we compare the distribution of NPMIs discovered by our SAE method, with a few other methods for discovering correlated feature pairs:

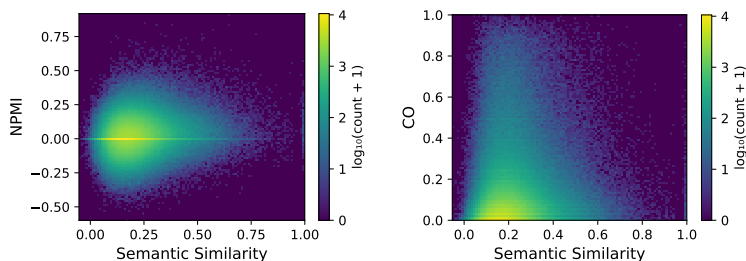


Figure A.46: Histogram of correlation metric (left: NPMP, right: CO) and semantic similarity of latent pairs. We choose NPMP as our metric.

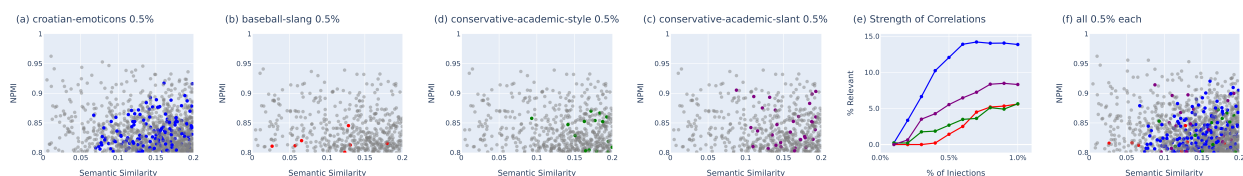


Figure A.47: (a)-(d) We plot the discovered group of pairs ($\text{NPMP} > 0.8$, semantic similarity < 0.2) for each type of text injected, with 0.5% of texts being injected texts. Relevant pairs are colored. (e) We show the proportion of relevant pairs in the candidate group for different injection levels 0.1%-1%. (f) We inject all 3 texts at once.

Injection	Latent 1 Relevant	Latent 2 Relevant
croatian-emoticons	croatian, russian, slavic	emoticon, emoji
baseball-slang	valley girl, slang, endearment	game, sport, baseball
conservative-academic_style	economic, political, business	academic, formal
conservative-academic_slant	economic, politic, business	communis, free, libert, interven, interfer

Table A.29: Keywords used to judge if a latent pair is relevant to the injected correlations.

1. **Random SAE baseline.** We randomly sample 100 SAE latent pairs (of sufficient frequency), relabel each and verify its presence in the dataset with an LLM, and compute the verified NPMP. We see that most randomly sampled pairs have low NPMP, as expected, showing that the SAE method of selecting pairs with high NPMP provides a strong signal.
2. **LLM baseline.** We prompt an LLM to identify meaningfully different feature correlations in the dataset:

```
You are given a dataset of {n_samples} documents.

Your task is to identify co-occurrences of meaningfully different features. A co-occurrence refers to when two features both appear WITHIN the same document.

Each feature can be:
- A topic, subject, concept, or idea
```

Injection	Injection Rate	No. of batches discovered
croatian-emoticons	0.2%	0/10
	0.5%	2/10
	1.0%	1/10
baseball-slang	0.2%	0/10
	0.5%	4/10
	1.0%	10/10
conservative-academic_style	0.2%	0/10
	0.5%	1/10
	1.0%	2/10
conservative-academic_slant	0.2%	1/10
	0.5%	3/10
	1.0%	6/10

Table A.30: For each type of injected correlation, at various injection rates, we count the number of batches where the LLM correctly identifies a related correlation.

```

- A specific language, style, tone, or sentiment
- A specific linguistic, rhetorical, or syntactic pattern
- Or any other identifiable textual property

We are interested in feature pairs that co-occur more than once across the dataset, i.e. the same feature pair co-occurs in multiple documents, even if only in a few documents.
We are only interested in feature pairs where the two features are meaningfully different. This means the two features cannot be trivially similar or extremely related.
Feature pairs can involve different feature types that co-occur, for example, between two semantically different concepts, or between a linguistic pattern and a concept, or between a linguistic and formatting pattern.
We are especially interested in feature co-occurrence pairs that are surprising, unexpected, interesting, or otherwise notable, even if this co-occurrence occurs only in a few documents.
Each feature in a pair should be described with a precise phrase that describes what the feature is about.

Return your answer as a JSON object with the following format, with up to 10 feature pairs:
{{
  "feature_pairs": [
    {{
      "feature_1": "feature_1_description",
      "feature_2": "feature_2_description"
    }},
    {{
      "feature_1": "feature_1_description",
      "feature_2": "feature_2_description"
    }},
    ...
  ]
}}

{"\n".join([f"---BEGIN DOC {i+1}---\n{text}\n---END DOC {i+1}---" for i, text in enumerate(sampled_texts)])}

```

We take the feature pairs generated by the LLM, verify each feature’s presence in the dataset with an LLM and compute the verified NPML.

3. **Correlated Topic Model (CTM).** We train a CTM [137, 142] to discover topics from word co-occurrences. We fix $n_{\text{topics}} = 100$ and consider a topic present in a document if it is among the top 5 topics in the document. This gives us the occurrences of the 100 discovered topics, from which we compute the verified NPML. The NPMLs tend to be low, even though the CTM allows for correlations between topics, suggesting that the CTM is not suited for discovering highly correlated topics.

We also report the distribution of conditional occurrence (CO) (see Appendix A.36) among the

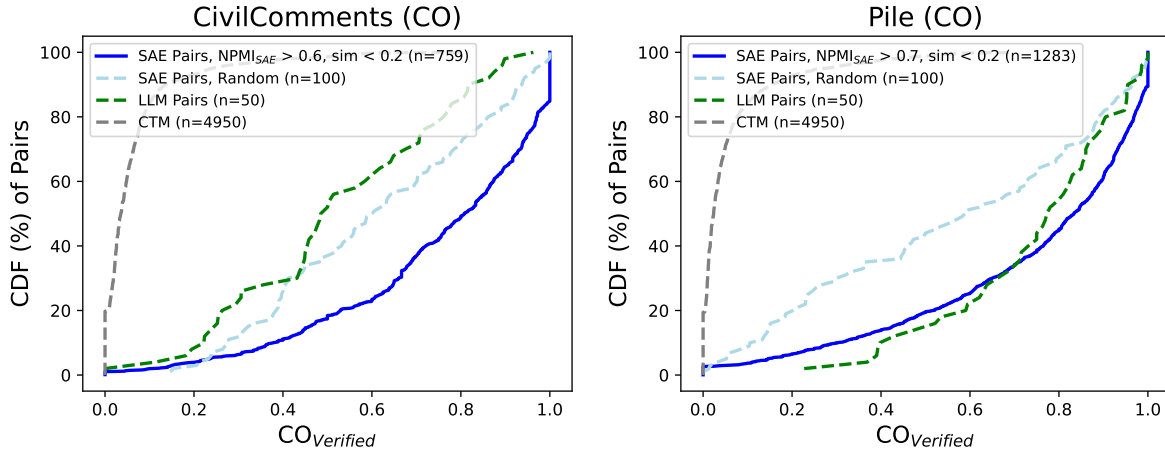


Figure A.48: CDF of conditional occurrence for pairs discovered by every method, for CivilComments (left) and the Pile (right).

discovered pairs for all methods (Figure A.48), to confirm that even when using a NPMI cutoff, the SAE method finds pairs with high CO and thus are “truly correlated” in some sense.

LLM hypotheses for real-world correlations. For each of the CivilComments (5k), Pile (5k) and Tulu (10k) datasets, we shuffle and split them into batches of 1k documents each. For each batch, we ask an LLM for up to 10 interesting hypotheses.

For CivilComments (Table A.31) and Pile (Table A.32), we verified the presence of each concept on a 1k sample from the same dataset. For Tulu (Table A.33), we note that the LLM baseline did not find the “math and hope” correlation and show 20 random samples of the 100 hypotheses.

Concept 1	Concept 2	NPMICO		$P(C_1 C_2)$		$P(C_2 C_1)$
Sarcastic or dismissive tone	Reference to Donald Trump's political actions or statements	0.422	0.617	0.617		0.123
Use of exclamation points for emphasis	Expression of strong negative emotion (e.g., anger, frustration)	0.567	0.675	0.328		0.675
Discussion of political figures or parties (e.g., Trump, Liberals, Republicans)	Accusations of lying, dishonesty, or manipulation	0.508	0.513	0.513		0.250
Critique of media bias or 'fake news'	Discussion of Russian interference in elections	0.000	0.000	0.000		0.000
Use of rhetorical questions	Challenge to an opposing viewpoint or argument	0.624	0.875	0.372		0.875
Discussion of religious beliefs or institutions	Critique of hypocrisy or inconsistency in actions versus stated beliefs	0.429	0.449	0.131		0.449
Reference to specific US states or cities (e.g., Alaska, Hawaii, Chicago)	Discussion of local governance or infrastructure issues	0.515	0.306	0.306		0.255
Discussion of environmental issues (e.g., climate change, pollution)	Skepticism or denial of scientific consensus	0.638	0.600	0.600		0.146
Use of informal or colloquial language	Expression of personal opinion or anecdote	0.827	0.862	0.813		0.862
Discussion of social justice issues (e.g., racism, equality)	Accusations of political correctness or 'virtue signaling'	0.453	0.583	0.583		0.047
Sarcastic tone	Critique of political figures (e.g., Trump, Trudeau)	0.501	0.471	0.471		0.298
Discussion of economic policy	Critique of government spending or taxation	0.637	0.482	0.363		0.482
Reference to 'fake news'	Critique of media bias	0.594	0.563	0.136		0.563
Use of rhetorical questions	Expression of skepticism or disbelief	0.620	0.649	0.449		0.649
Discussion of environmental issues	Critique of government inaction or corporate responsibility	0.493	0.357	0.147		0.357
Critique of political correctness	Defense of free speech or traditional values	0.590	0.222	0.222		0.182
Discussion of gun control	Arguments for or against gun ownership rights	0.786	0.500	0.500		0.455
Religious references or analogies	Critique of institutional religion or religious hypocrisy	0.799	0.963	0.963		0.361
Discussion of social inequality (e.g., poverty, racism)	Critique of societal structures or government policies	0.488	0.761	0.148		0.761
Use of informal or colloquial language	Expression of strong personal opinion or frustration	0.860	0.893	0.846		0.893
Sarcastic tone	Critique of political figures or parties	0.517	0.437	0.437		0.399
Discussion of economic policy	Criticism of government spending or taxation	0.628	0.444	0.374		0.444
Use of rhetorical questions	Expression of strong disagreement or disbelief	0.584	0.815	0.348		0.815
Critique of media bias	Accusations of 'fake news' or propaganda	0.604	0.458	0.458		0.193
Focus on social issues (e.g., immigration, healthcare)	Attribution of blame to specific political ideologies (e.g., 'left' or 'right')	0.374	0.238	0.238		0.128
Informal language or slang	Direct address to other commenters	0.515	0.772	0.772		0.243
Religious references or arguments	Critique of societal morality or values	0.453	0.455	0.157		0.455
Discussion of environmental issues	Skepticism towards scientific consensus or government initiatives	0.507	0.432	0.148		0.432
Personal anecdotes or experiences	Generalizations about groups of people (e.g., 'millennials', 'conservatives')	0.349	0.263	0.150		0.263
Hyperbolic language	Prediction of negative future outcomes	0.504	0.715	0.715		0.199
Sarcastic tone	Critique of political figures or policies	0.543	0.505	0.420		0.505
Use of rhetorical questions	Discussion of political or social issues	0.539	0.706	0.320		0.706
Ad hominem attacks	Discussion of political figures	0.498	0.477	0.248		0.477
Discussion of Trump's presidency	Accusations of lying or dishonesty	0.283	0.113	0.088		0.113
Use of all caps for emphasis	Expression of strong emotion or outrage	0.513	0.779	0.187		0.779
Analogy or metaphor	Critique of a complex system or situation	0.473	0.707	0.135		0.707
Discussion of media bias	Accusations of 'fake news'	0.511	0.294	0.294		0.116
Reference to historical events or figures	Comparison to current political situations	0.372	0.252	0.146		0.252
Discussion of economic issues	Critique of government spending or taxation	0.678	0.642	0.642		0.391
Use of profanity or vulgar language	Expression of strong disapproval or contempt	0.413	0.898	0.077		0.898
Criticism of Donald Trump's character or policies	Use of informal or derogatory language	0.480	0.817	0.111		0.817
Discussion of political parties (Democrats/Republicans, Liberals/Conservatives)	Accusations of hypocrisy or inconsistency	0.427	0.254	0.206		0.254
Mentions of 'fake news' or media bias	Sarcasm or ironic tone	0.395	0.744	0.060		0.744
Arguments about gun control or gun violence	Exaggerated or hyperbolic statements	0.278	0.450	0.023		0.450
Critique of government spending or economic policy	Call for accountability or transparency	0.388	0.224	0.211		0.224
Religious or moral arguments	Critique of specific religious institutions or leaders	0.733	0.806	0.806		0.301
Discussion of immigration or refugee issues	Accusations of racism or xenophobia	0.457	0.182	0.182		0.143
References to historical events or figures	Drawing parallels to current political situations	0.353	0.306	0.119		0.306
Concerns about environmental issues or climate change	Skepticism towards scientific consensus or political motives	0.516	0.485	0.128		0.485
Discussion of social justice issues (e.g., racism, gender equality)	Personal anecdotes or appeals to personal experience	0.312	0.195	0.126		0.195

Table A.31: Hypothesized correlations in CivilComments generated by LLM.

Concept 1	Concept 2	NPMICO	P(C ₁ C ₂)	P(C ₂ C ₁)
Discussion of specific programming language features or issues (e.g., Python, JavaScript, C#)	Question-and-answer format typical of programming forums	0.836	0.750	0.742
Technical discussion of software development or IT infrastructure	Question-and-answer format typical of programming forums	0.828	0.953	0.953
Use of code snippets to illustrate programming concepts	Question-and-answer format typical of programming forums	0.743	0.621	0.621
Medical research or clinical study findings	Detailed scientific or medical terminology	0.869	0.986	0.670
Discussion of specific medical conditions or treatments	Detailed scientific or medical terminology	0.794	0.953	0.511
Geographic or demographic data analysis	Statistical analysis or quantitative findings	0.527	0.905	0.079
Discussion of specific historical figures or events	Biographical information	0.679	0.535	0.414
Discussion of specific geographic locations or regions	Cultural or historical context	0.647	0.731	0.731
Analysis of financial markets or economic trends	Discussion of specific companies or industries	0.582	0.773	0.121
Discussion of specific software or platforms (e.g., WordPress, Magento)	Instructions or advice on configuration/usage	0.800	0.830	0.591
Discussion of specific programming language features (e.g., C#, Python, Java, Javascript, SQL, PHP, Swift, Objective-C, R, Go, Perl, F#, VB.NET, C++)	Question and Answer format	0.669	0.662	0.394
Discussion of specific programming language features (e.g., C#, Python, Java, Javascript, SQL, PHP, Swift, Objective-C, R, Go, Perl, F#, VB.NET, C++)	Code snippets provided as examples or solutions	0.841	0.850	0.654
Medical research study or clinical trial	Focus on specific diseases or conditions (e.g., cancer, diabetes, neurological disorders)	0.818	0.860	0.860
Medical research study or clinical trial	Quantitative data or statistical analysis	0.746	0.638	0.617
Medical research study or clinical trial	Use of specialized medical terminology	0.935	0.980	0.826
Discussion of specific programming language features (e.g., C#, Python, Java, Javascript, SQL, PHP, Swift, Objective-C, R, Go, Perl, F#, VB.NET, C++)	Error messages or debugging scenarios	0.663	0.684	0.684
Discussion of specific programming language features (e.g., C#, Python, Java, Javascript, SQL, PHP, Swift, Objective-C, R, Go, Perl, F#, VB.NET, C++)	Reference to external libraries or frameworks (e.g., jQuery, Android Studio, Spark, Django, React Native)	0.700	0.701	0.701
Medical research study or clinical trial	Animal models used in research	0.701	0.951	0.951
Medical research study or clinical trial	Focus on specific biological mechanisms or pathways	0.748	0.763	0.763
Medical research study or clinical trial	Use of imaging techniques (e.g., MRI, ultrasound, scintigraphy)	0.521	0.810	0.810
Discussion of specific programming language features (e.g., Func, IEnumerable<char>, instance)	Problem-solving in a Q&A format, often involving debugging or optimizing code snippets	0.716	0.797	0.355
Technical documentation or code comments related to software development (e.g., copyright notices, license information, API descriptions)	Mentions of specific software frameworks, libraries, or tools (e.g., Spring-boot, Netty, Vue.js, React Native)	0.478	0.397	0.206
Medical research focusing on specific diseases or conditions (e.g., Hodgkin disease, prostate cancer, diabetes, epilepsy)	Detailed descriptions of biological mechanisms, physiological processes, or pharmacological interventions (e.g., gene expression, hormone response, fatty acid metabolism)	0.674	0.591	0.431
Discussion of specific geographical locations or regions (e.g., North Carolina, Japan, Australia, Texas)	Mentions of historical events, political figures, or cultural aspects related to those locations (e.g., 1844 United States presidential election, Prime Ministers of Japan, Deepwater Horizon oil spill)	0.678	0.860	0.860
User-generated content in a Q&A format, often seeking technical solutions (e.g., How to, Is it possible to)	Code snippets or examples provided as part of a question or answer, demonstrating a technical problem or solution	0.712	0.630	0.630
Descriptions of physical products or consumer goods (e.g., turntable, boots, space heater, refrigerator)	Emphasis on features, specifications, or benefits of the product, often with marketing language (e.g., Key features, Appealing look, High Capacity-Size Ratio)	0.794	0.750	0.750
Legal or judicial proceedings, including court cases and appeals (e.g., United States Court of Appeals, Supreme Court of Florida)	Mentions of specific legal documents, acts, or concepts (e.g., Civil List Act, Controlled Substances Act, currency exception)	0.647	0.429	0.250
Scientific research papers or abstracts detailing experimental methods and results (e.g., Purification and characterization, Effects of amide constituents, Quantitation of 20-hydroxy-5,8,11,14-eicosatetraenoic acid)	Use of specialized scientific terminology and acronyms (e.g., HPLC, TLC, NMR, ELISA, qPCR)	0.941	0.955	0.955
Discussions about web development technologies and issues (e.g., URL encoding, CSS, JavaScript, HTML)	References to specific web browsers or platforms (e.g., Chrome, Safari, iOS, Android)	0.461	0.390	0.390
Content related to music, artists, or albums (e.g., Jimi Hendrix, Harry Styles, Blackjack, Babymetal)	Mentions of specific songs, track listings, or musical genres (e.g., Imperial Blaze, Worlds Apart, Gimme Chocolate)	0.863	1.000	1.000
Discussion of specific programming language features or syntax (e.g., Python, Java, C#, JavaScript)	Question-and-answer format for technical problem-solving	0.780	0.715	0.624
Medical research or clinical study findings	Focus on specific biological mechanisms or pathways (e.g., proteins, genes, cells)	0.754	0.780	0.780
Geographical or place-name disambiguation	Wikipedia-style entry or factual description of a place	0.635	0.476	0.476
Use of specific technical terms or jargon (e.g., disambiguation, phylogenetic, electroluminescence)	Scientific or academic research paper abstract	0.675	0.867	0.867
Discussion of software development tools or environments (e.g., Git, Eclipse, Visual Studio)	Code snippets or programming examples	0.631	0.595	0.347
Analysis of political or social issues	Quoted statements or opinions from individuals or organizations	0.590	0.522	0.214
Description of a product or service	Marketing or promotional language	0.718	0.769	0.769
Discussion of specific cultural or entertainment media (e.g., movies, TV shows, music)	Personal opinion or commentary on the media	0.584	0.392	0.307
Legal or court-related document	Formal, structured language typical of legal texts	0.877	0.854	0.636
Discussion of specific scientific concepts (e.g., physics, chemistry, biology)	Mathematical equations or formulas	0.199	0.227	0.227
Discussion of specific programming language features or syntax (e.g., Python and operator, C# generics, PHP sessions)	Question-and-answer format, often from a technical forum like Stack Overflow	0.758	0.825	0.469
Medical research or clinical study focusing on a specific disease or treatment (e.g., cancer, diabetes, specific drug effects)	Detailed scientific terminology and methodology (e.g., randomized trial, pharmacokinetics, immunohistochemistry)	0.789	0.895	0.523
Legal documents or court case summaries, often with citations and formal language	Mentions of specific legal entities, jurisdictions, or case names (e.g., Supreme Court of North Carolina, United States Court of Appeals)	0.900	0.947	0.947
Technical specifications or code snippets related to software development (e.g., Dockerfile, Javascript, XML configuration)	Copyright notices or licensing information (e.g., GNU General Public License, Apache License)	0.523	0.818	0.818
Travel or tourism-related content, often describing destinations or experiences	Personal anecdotes or first-person narratives about travel	0.707	0.714	0.714
Discussion of specific hardware components or technical devices (e.g., smartphone, printer, sensor, computer graphics)	Problem-solving or troubleshooting context (e.g., inconsistent results, cannot be detected, error messages)	0.441	0.369	0.195
Content related to food, recipes, or culinary topics	Mentions of specific ingredients or cooking methods	0.810	0.882	0.882
Descriptions of geographical locations (e.g., cities, countries, regions)	Categorization or metadata related to geography (e.g., Category:Populated places, Category:Mountains)	0.674	0.985	0.985
Discussion of art, artists, or creative works (e.g., paintings, films, music)	Personal opinions or subjective evaluations of the art	0.793	0.898	0.898
Content related to sports or athletic activities (e.g., football, cycling, basketball)	Mentions of specific teams, athletes, or events	0.922	1.000	1.000

Table A.32: Hypothesized correlations in the Pile generated by LLM.

Feature 1 Description (User Request)	Feature 2 Description (System Response)
Question about a specific technical concept or tool	Explanation of the concept or tool with examples
Request for a definition/explanation	Detailed explanation of a concept
Mathematical problem involving calculus or differential equations	Step-by-step solution using symbolic math (SymPy) or numerical methods (NumPy, SciPy) in Python
Request for a Python function to calculate a sum or average from a list of numerical data	Python code defining a function that iterates through the list and performs the requested aggregation
Request for translation (non-English to English)	English translation of the provided text
Mathematical word problem with multiple steps	Step-by-step solution with intermediate calculations
Request for Python function to perform statistical calculations (e.g., average, correlation)	Python function using <code>numpy</code> for statistical operations
Request for a programming problem with an erroneous code snippet	Identification and correction of errors, followed by a correct implementation
Request for a Python function to perform string manipulation (e.g., palindrome check, word count)	Python code defining a function that uses string methods and loops for text processing
Question about a specific entity or concept	Direct answer or explanation of the entity/concept
Request for a Python function to handle data structures (e.g., nested lists, dictionaries)	Python code demonstrating iteration and access patterns for complex data structures
Request for Python function to manipulate strings or lists	Python function using string methods like <code>split()</code> , <code>join()</code> , <code>lower()</code>
Request for a detailed explanation of a technical concept	Comprehensive explanation of the concept with examples or analogies
Request for a code snippet in one language (e.g., Python, Java)	Equivalent code snippet in another specified language (e.g., C#, JavaScript, Swift) with explanatory comments
Request for a Python function to filter or categorize data based on conditions	Python code defining a function that uses conditional logic and list/dictionary manipulation to filter/categorize
Request for a creative story or scenario	Detailed narrative with character descriptions and plot development
Request for a JSON output	JSON object as output
Mathematical problem solving	Step-by-step mathematical derivation
Request for translation (English to non-English)	Non-English translation of the provided text
Request for a JavaScript function to manipulate DOM elements or data	JavaScript code snippet using DOM manipulation or array methods

Table A.33: Sample of 20 Tulu hypotheses generated by LLM.

A.37 Additional Results—Clustering

Experiment setup

To filter for latents relevant to a query, we can find latents whose labels’ dense embeddings are the most similar (e.g. top $k = 100$) to that of a provided keyphrase. Multiple keyphrases can be provided and the union of all these latents taken, which would effectively ignore other unrelated latents.

We can optionally use an LLM to help with keyphrase generation given a query, e.g. “I want to cluster by news topic” would require latents related to all possible relevant keyphrases (“sports”, “politics”...) which the LLM can generate. The prompt we used is:

```
system_prompt = """
You are an NLP feature-brainstorming assistant.

Task: Given a user query, suggest 2 to 5+ distinctive and semantically specific keywords or phrases that capture the key concepts relevant to that query.

- If the goal refers to a binary or low-dimensional axis (e.g. sentiment, tense, polarity), return only the most salient few items (2-4).
- If the axis is broad or multi-class (e.g. topic, genre, domain), return more diverse sub-categories (up to 10).
- Each item should be a single coherent concept that could plausibly describe the activation of a sparse autoencoder feature.
- Include contrasting pairs or subtypes when applicable (e.g. "positive", "negative").
- Avoid generic catch-alls like "style", "content", or "other".
- Return each item on its own line, without bullets or numbering.
"""

true_label_col_to_user_query = {
    "sentiment": "I have a dataset of news articles. I want to cluster them based on the sentiment of the article.",
    "temporal": "I have a dataset of news articles. I want to cluster them based on the temporal framing of the article.",
    "topic": "I have a dataset of news articles. I want to cluster them based on the main topic of the article.",
    "style": "I have a dataset of news articles. I want to cluster them based on the writing style of the article."
}
```

Generating cluster labels. For a cluster, we can find the top five latents by diffing the cluster with all texts outside the cluster. We also find the top five examples with the highest affinities to the rest of the cluster as the top “central” examples. We do this for each cluster, then generate distinctive cluster labels using the following prompt:

```
system_prompt = """
You are an assistant for labeling clusters of natural language text.
You will be given multiple clusters at once. For each cluster, you have the top {n_relabel} distinctive features and top {n_relabel} examples.
Your task is to create DISTINCTIVE, human-like labels that capture what unites each cluster.

IMPORTANT:
- Each cluster label must be DIFFERENT from all others
- Focus on what makes each cluster UNIQUE, not just common themes
- Create natural, descriptive labels that a human would understand immediately
- Labels can be longer and more detailed if needed to capture the essence
- Look for patterns in content, tone, style, intent, or context
- Only quote specific phrases if they're extremely clear and defining
- If a cluster is truly unclear, label it "UNCLEAR"

Return your response in this exact format:
Cluster 0: [label]
Cluster 1: [label]
Cluster 2: [label]
...and so on

Return ONLY the cluster labels in this format, no other text.
"""
```

Ground truth evaluation

We generate news paragraphs with four independent “axes of variation”: 1. topic (health, technology, sports, politics), 2. sentiment (positive, negative), 3. temporal framing (focusing on past, present or future) and 4. writing style (factual or narrative). We query an LLM for keyphrase

generation, saying that we want to cluster by each of the four axes, and keeping the union of the the top $k = 100$ latents most similar to each keyphrase. The SAE can separate this synthetic dataset well along different axes (Figure A.49).

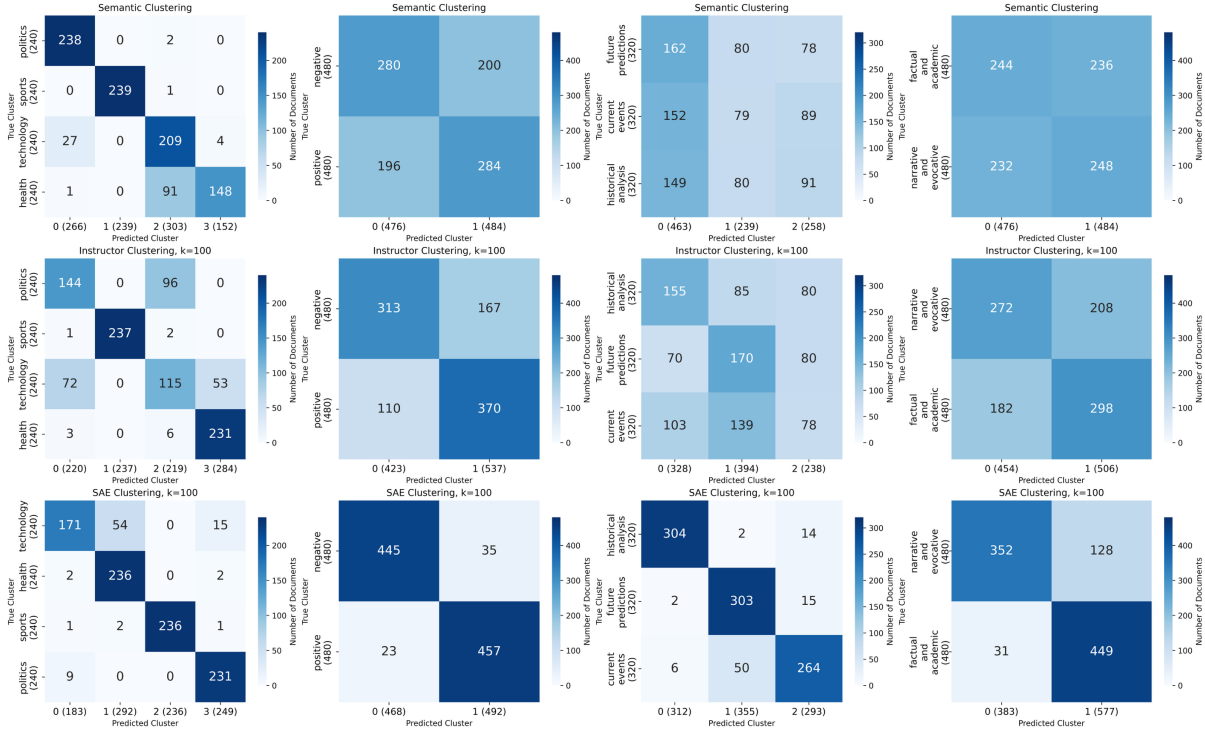


Figure A.49: Dense embedding (top row), instruction-tuned embedding (middle row) and SAE embedding (bottom row) clustering results: (1) topic (2) sentiment (3) temporal framing and (4) writing style. Mappings from clusters to true labels are chosen with the Hungarian algorithm [135].

Real-world evaluation—IMDb

Similarly to the GSM8k dataset, we cluster IMDb movie descriptions using SAE embeddings. Using the full embedding with all latents, we find clusters of *how* the descriptions are written, providing additional insight compared to the genre-based dense embedding clustering (Figure A.50a).

With targeted clustering, the SAE can cluster by e.g. “how the characters are described”, giving a new set of clusters. The instruction-tuned embedding still biases towards clustering by genre despite the instruction (Figure A.50b).

Real-world evaluation—accuracy

We show the per-cluster accuracies for dense embedding and SAE clustering, on ChatbotArena prompts, responses and the Pile, in Figure A.51. We see that the SAE clusters have comparable per-cluster accuracies with embeddings, with generally higher variance across clusters. This suggests the clusters are similarly valid—the SAE indeed groups similar texts together.

Dense embedding				SAE embedding: all latents			
Cluster LLM Label	Top Example	Acc.	Z	Cluster LLM Label	Top Example	Acc.	Z
Series of Soldiers and Conflict in Wartime	In Nazi-occupied France during World War II, a plan to assassinate Nazi leaders by a group of Jewish U.S. soldiers coincides with a theater owner's vexatious plans for the same.	54.4	-16.8	Movie plot summaries starting with "A..." or "An..."	A college professor bonds with an abandoned dog he takes into his home.	0.544	-16.8
Unlikely Bonds and Romantic Connections	A failed movie star and a neglected young woman form an unlikely bond after crossing paths in Tokyo.	19.6	-27.2	Movie plot summaries about duos or small groups, starting with a number	Two newly paired cops who are complete opposites must put aside their differences in order to catch a gang of drug smugglers.	0.196	-27.2
Science Fiction and Fantasy Adventure Movie Plots	A troubled child summons the courage to help a fatally ill alien escape Earth and return to his home world.	79.1	-18.7	Movie plot summaries that start by naming the main character	Blacksmith Will Turner teams up with eccentric pirate "Captain" Jack Sparrow to save his love, the governor's daughter, from Jack's former pirate allies, who are now undead.	0.791	-18.7
Character-Driven Drama about Personal and Familial Struggles	A young man in a small Midwestern town struggles to care for his mentally-disabled younger brother and murderously obese mother while attempting to pursue his own happiness.	77.8	-18.0	Movie plot summaries that begin by establishing the setting or time period	In 1986 Italy, romance blossoms between a sciency-year-old student and the older man hired as his father's research assistant.	0.778	-18.0
Crime, Heist, and Detective Thriller Movie Plots	A police detective, a bank robber, and a high-power broker enter high-stakes negotiations after the criminal's brilliant heist spirals into a hostage situation.	87.3	-32.6	Movie plot summaries beginning with the phrase "The story of..."	The story of a team of female African-American mathematicians who served a vital role in NASA during the early years of the U.S. space program.	0.873	-32.6

Instruction-tuned embedding: "Represent the text so I can cluster them by the way the characters are described."				SAE embedding: "I want to cluster by how the characters are described."			
Cluster LLM Label	Top Example	Acc.	Z	Cluster LLM Label	Top Example	Acc.	Z
Character-driven dramas about a man's personal journey or crisis	A teacher lives a lonely life, all the while struggling over his own custody. His life slowly gets better as he finds love and receives good news from his son, but his new luck is about to be brutally shattered by an innocent little lie.	0.872	-14.8	Character-driven dramas about a man's personal journey or crisis	A teacher lives a lonely life, all the while struggling over his own custody. His life slowly gets better as he finds love and receives good news from his son, but his new luck is about to be brutally shattered by an innocent little lie.	0.355	-5.73
Action-adventure plots about a group teaming up against a common threat	A dashing thief, his gang of desperados and an intrepid policeman struggle to find a prince from an evil count's clutches, and learn the hidden secret to a fabulous treasure that the hold part of a key to.	0.655	-14.2	An individual's journey into a new or mysterious situation	In a future world devastated by disease, a convict is sent back to time to gather information about the man-made virus that wiped out most of the human population on the planet.	0.335	-4.51
War movie plots centered on the experiences of soldiers in historical conflicts	April 6th, 1917: As a regiment assembles to wage war deep in enemy territory, two soldiers are assigned to race against time and deliver a message that will stop 1,600 men from walking straight into a deadly trap.	0.340	-14.3	Narratives examining a specific event, organization, or time period	An examination of the machinations behind the scenes at a real estate office.	0.714	-1.03
Crime and thriller plots involving detectives, heists, and criminal pursuits	A private detective hired to expose an adulterer finds himself caught up in a web of deceit, corruption, and murder.	0.850	-31.4	A group of people uniting for a common purpose	Los Angeles citizens with rarely separate lives collide in interweaving stories of race, loss and redemption.	0.171	-0.627
Romantic plots centered on complicated or unlikely relationships	With the intention to break free from the strict familial restrictions, a suicidal young woman sets up a marriage of convenience with a forty-year-old addict, an act that will lead to an outburst of serious love.	0.486	-26.4	Stories centered on the dynamic between an unlikely pair of characters	A recently laid off factory worker kidnaps his former boss' friend's daughter, hoping to use the ransom money to pay for his sister's kidney transplant.	0.403	-15.2

(a) Normal clustering with dense embeddings [left] and the full SAE embedding [right]. The SAE embedding clusters along how the description is written, with generally good cluster accuracy.

(b) Targeted clustering with instruction-tuned embeddings [left] and the reduced SAE embedding [right]. The SAE embedding finds clusters of character descriptions.

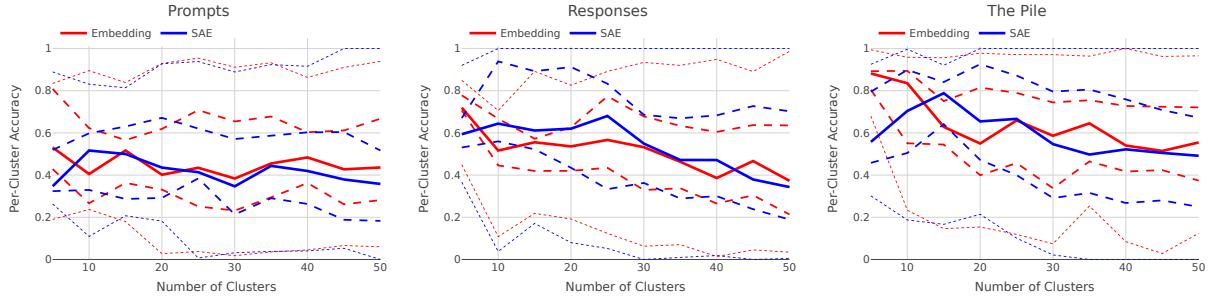


Figure A.51: Per-cluster accuracies for different n_{clusters} for prompts, responses and the Pile. The solid lines are the median, dashed lines the interquartile range and dotted lines the range.

Dense Embedding	Acc.	SAE Embedding	Acc.
Simple "Hello World" style Python code requests	0.060	Simple "What is the capital of..." questions	0.000
Debating who is the best athlete	0.173	Single-topic prompts	0.089
Questions about numbers and their properties	0.274	Requests to write a poem in German	0.179
Questions and stories about cats and dogs	0.313	"What is..." questions in German	0.237
Math word problems involving time and rates	0.412	Informal greetings in English and Spanish	0.345
Food recipes and cooking instructions	0.482	Probing the AI's knowledge on specific topics	0.387
Generating video game ideas and recommendations	0.613	Questions and requests in Russian and Polish	0.448
Questions about world history and historical events	0.695	Writing Python scripts for specific tasks	0.529
Simple greetings and conversation starters	0.797	Complex instructions for AI reasoning and persona before a task	0.798
Requests for jokes	0.939	Recommending films similar to specific video games	1.000

Table A.34: Example clusters from ChatbotArena prompts.

We show qualitative examples of cluster descriptions in Tables A.34-A.36. Since these datasets are highly diverse, we show results from $n_{\text{clusters}} = 50$, randomly sampling one cluster per accuracy quantile. In these cases, the SAE cluster descriptions are similar in style to semantic cluster descriptions.

Failure to recover ground truth labels for sentiment and emotion clustering. Since SAEs are not trained to represent similarity, we may not obtain "desired" clusters for a dataset

Dense Embedding	Acc.	SAE Embedding	Acc.
Defining "Machine Learning"	0.034	Presenting Code Snippets on Request	0.005
Stating the Current US President	0.123	Concise Answers to Factual Questions or Riddles	0.082
Inappropriate or Sexually Suggestive Narratives	0.211	Discussing Philosophical and Abstract Concepts	0.164
Solving Basic Algebraic Equations	0.263	Generating Numbered Lists of Items	0.256
Standard Assistant Greeting	0.333	Repetitive or Malformed Lists and Text	0.302
Business and Workplace Productivity Strategies	0.391	Step-by-Step Recipes and Workout Plans	0.391
Numbered Lists of Self-Help and Wellness Advice	0.596	Explanations in German	0.589
Solving Simple Math Word Problems	0.646	Simple Arithmetic Calculations	0.714
Providing Code Snippets	0.770	Original Poetry with Rhyme and Meter	0.907
Assistant Expressing Confusion and Requesting Clarification	0.987	Identifying Capital Cities	1.000

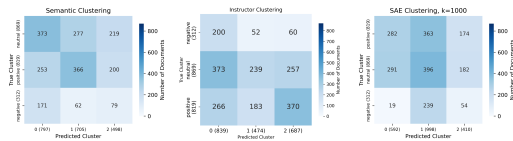
Table A.35: Example clusters from ChatbotArena responses.

Dense Embedding	Acc.	SAE Embedding	Acc.
Unity Engine Asset Metadata Files	0.124	Abbreviated or Incomplete User Input	0.000
Research Abstracts on Molecular Biology and Genetics	0.320	Event Announcements and Local News Snippets	0.073
Celebrity Gossip and Lifestyle Articles about Female Public figures	0.370	jQuery and JavaScript Code Debugging and Implementation Questions	0.236
Research Abstracts on Chemical Synthesis and Characterization	0.418	Extremely Brief and Ambiguous User Inputs	0.333
JavaScript/Node.js Modules and Configuration Files	0.488	News Reports on Political and Social Events	0.408
Short Biographies of Politicians and Public Figures	0.569	Technical Q&A with Code and System Configuration Issues	0.554
Scientific Abstracts on Cognitive Neuroscience and Neuropsychology	0.623	Programming Language Test Files and Boilerplate Code	0.619
Short Biographies of Athletes	0.746	Federal Court of Appeals Case Citations	0.731
Clinical Studies on Surgical Procedures and Outcomes	0.847	Advanced Mathematical Problem Solving and Proofs	0.944
US Federal Court Case Filings and Orders	0.966	Zoological Species Descriptions	1.000

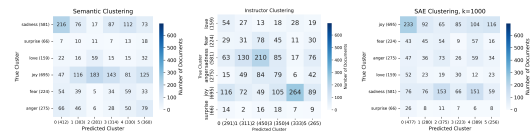
Table A.36: Example clusters from The Pile.

with ground-truth cluster labels. For instance, if an SAE has learned many more “sadness” latents than “surprise” latents, clustering may distinguish between different types of “sadness” more than between “sadness” and “surprise”.

Figures A.52a and A.52b show the failure of both embedding baselines and SAEs to align exactly with ground truth labels. While finetuned models (for sentiment/emotion) do achieve good performance on these tasks, we do not expect these general purpose embedding baselines to align with ground-truth labels. For our SAE method, we were unable to find a good combination of queries and k .



(a) Twitter sentiment [224] clustering results.



(b) Twitter emotion [231] clustering results.

A.38 Additional Results—Retrieval

Example queries. We show 5 selected queries for each dataset to illustrate the types of properties we aim to look for in Table A.37.

Retrieval baselines. We describe each method in detail in Table A.38.

For the BM25 baseline, we expand the query using the following:

```
prompt = f" ""
```

Query	
Prompts	1. unfiltered: The user requests or tries to trick the model to bypass or disable its built-in safety and content filters.
	2. meta: The user explicitly asks about the model itself, including its architecture, training data, internal workings, limitations, performance, capabilities, or guidelines.
	3. ethical-dilemma: The user explicitly poses an ethical or moral dilemma, scenario, or thought experiment.
	4. opinion: The user explicitly asks the model for its personal opinion, subjective judgment, or preference on a given topic.
	5. expert-role-simulation: The user explicitly instructs the model to respond from the perspective of a recognized expert, specialist, professional, or authoritative figure on a particular topic.
Responses	1. offensive-success: The model explicitly generates questionable, offensive, discriminatory, violent, sexually explicit, hateful, derogatory, or profane content.
	2. cot: The model explicitly demonstrates its reasoning or thought process in clear, sequential steps, outlining the logical progression leading to its conclusion or answer.
	3. roleplay-persona: The model responds from the point of view of, adopts, simulates, or maintains a specific persona, role, character, identity, or professional perspective in its response.
	4. disclaimer-warning: The model explicitly includes a disclaimer, warning, or caution, advising the user to consult a professional or that the information is not a substitute for expert advice (e.g., 'I am not a medical professional', 'This is not financial advice').
	5. empathy: The model explicitly expresses empathy, sympathy, understanding, compassion, emotional support, or validation toward the user's feelings, emotions, or experiences.
Reasoning Traces	1. similar: The model mentions or draws parallels to a similar or related problem it knows about, suggesting the same solution technique might apply.
	2. intuition: The model references using its intuition or gut feeling to make a guess or estimate, rather than relying purely on formal logic.
	3. idk: The model explicitly admits it lacks information.
	4. identifying-a-trap: The model explicitly identifies a potential 'trap', a common misconception, or a subtle aspect of the problem that could easily lead to an incorrect answer.
	5. edge-case: The model considers an edge case, special case, or boundary condition (such as zero, infinity, or maximal values) to check solution robustness.
The Pile	1. fan: The text references or discusses characters, settings, or events from a known fictional universe (e.g., Marvel, Star Wars, Harry Potter).
	2. changelog: The text lists software or document version updates, typically in bullet point or release-note format with dates or version numbers.
	3. email-letter-format: The model structures its response in the format of an email or a formal/informal letter, such as including elements like a salutation ('Dear...'), a body, and a closing ('Sincerely,...').
	4. popup-ads: The text includes pop-up advertisements or other promotional content that appears unexpectedly or does not fit the context of the surrounding text.
	5. hate-speech: The text expresses explicit hostility, slurs, or dehumanizing language targeted at a group based on race, gender, religion, sexuality, or other identity.
Biology Abstracts	1. human-trial: The abstract mentions the use of human or clinical trials.
	2. proteomics: The abstract mentions the generation, analysis or study of protein data.
	3. computational-biology: The text describes a study primarily based on computational models, algorithms, or simulations applied to biological data.
	4. negative-result: The abstract reports negative results, or a failure to achieve the expected outcome.
	5. mechanistic: The abstract mentions uncovering or explaining the underlying biological mechanism of a process, pathway, or phenomenon.
Short Stories	1. dystopian: The story is set in a dystopian or oppressive world.
	2. amnesia: The story includes a character suffering from memory loss, memory gap, or unable to remember their past or what happened.
	3. cheerful_dark: The story or protagonist is light-hearted or whimsical even in the midst of dark, violent, or tragic events.
	4. fourth-wall: The story includes breaking the fourth wall, commenting on its own nature as a work of fiction, or addressing the reader directly.
	5. archaic_language: The story includes archaic old-fashioned language, such as archaic words, phrases, or grammatical structures, often to evoke a specific time period.

Table A.37: Example queries across the six datasets.


```

I have a dataset of {type_of_text}, and I want to search among it for texts that fulfill a specific query.

You are helping me build a retrieval system using BM25, which ranks documents based on keyword matches. Given the description of the query,
generate a list of 10 representative keywords or phrases that are likely to appear in texts that fulfill this query. Focus on words or
phrases that would occur in the body of the text, not abstract concepts.

Return the list of keywords as a JSON list of strings.

QUERY: {query_string}
"""

```

For the OpenAI+LLM and Gemini+LLM baselines, we generate example phrases using:

```

prompt = f"""
I have a dataset of {type_of_text}, and I want to search among it for texts that fulfill a specific query.

The query is a description of a property. Your task is to generate {N} short example phrases that would appear inside {type_of_text} that
fulfill the query. Each phrase should show the desired behavior.

Do not repeat the query. Write "each phrase" as if they were part of the {type_of_text}.

Return the phrases as a JSON list.

QUERY: {query_string}
"""

```

LLM reranking of latents. For selection and reranking of latents that are relevant to a user query, we use the following prompt:

```

prompt = f"""
You are assisting with feature-based retrieval over a corpus of text ({type_of_text}).
You are given:

- A retrieval query describing a property of the texts we want to retrieve.
- A list of feature indices with their descriptions.

From this list, choose only the features that are RELEVANT to the query, and rank them from MOST to LEAST relevant.
Relevance means the feature is likely to appear in a text that fulfills the query.

### QUERY:
{query_string}

### FEATURES:
{'\n'.join(feature_descs)}

### OUTPUT FORMAT:
Return ONLY a list of relevant, reranked feature indices, in a valid JSON list, e.g. [14826, 481, 2310].
Make sure your features are a subset of the original features.
"""

```

Metrics. The formulae for the metrics we report are:

$$AP = \frac{1}{|R|} \sum_{k=1}^N \frac{|\{d_i \in R \mid i \leq k\}|}{k} \cdot \mathbf{1}\{d_k \in R\}$$

$$P@K = \frac{1}{K} \sum_{k=1}^K \mathbf{1}\{d_k \in R\}$$

MP@50. We report MP@50 across different methods and datasets, which may be more important to a practitioner as they are concerned with top results.

Hyperparameter dependence. We plot the dependence of MAP and MP@50 on the number of phrases used for query expansion (Figures A.54a-A.55a), and on temperature for latent aggregation (Figure A.55b). The performance of the SAE method is sensitive to the temperature. Aggregation is necessary as shown by the poor performance of $T = 0.01$ across datasets, due to labels being fine-grained and imprecise. We see in Figure A.55b that a higher T is better for

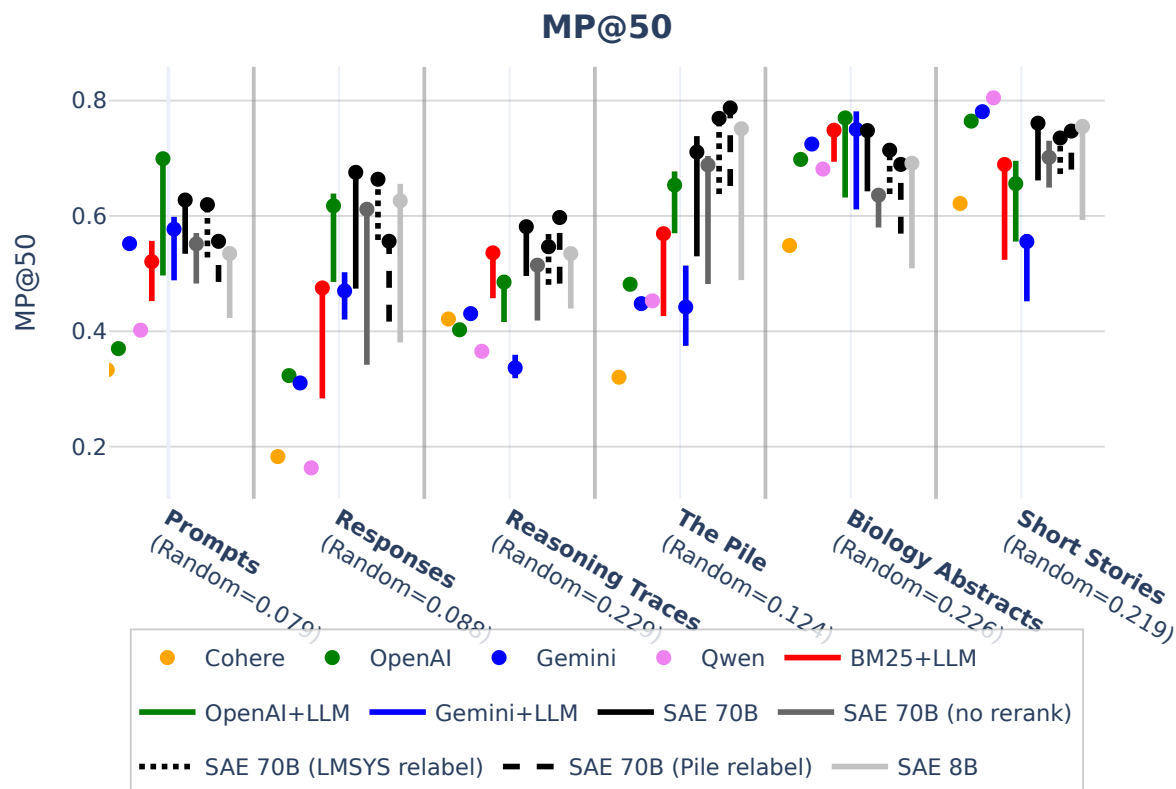
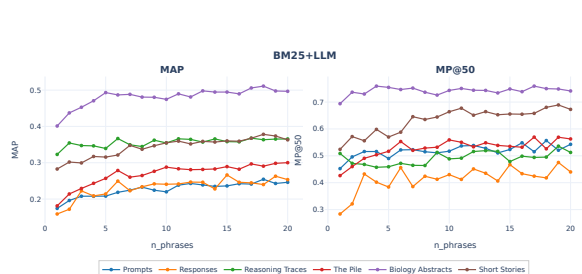
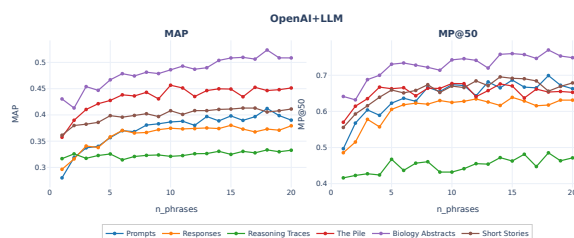


Figure A.53: MP@50 averaged over queries for each method and dataset. Query expansion uses 1–20 phrases; temperature varies from 0.01 to 1.5.

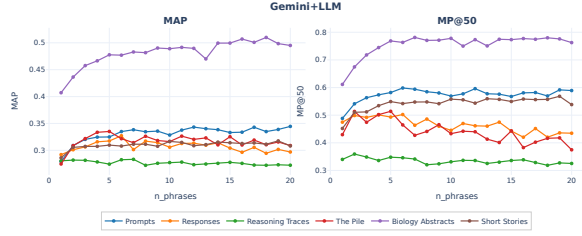
responses and the Pile, likely because the SAE was trained on chat data, thus it learned many higher-quality latents for that distribution that can be aggregated for overall better performance.



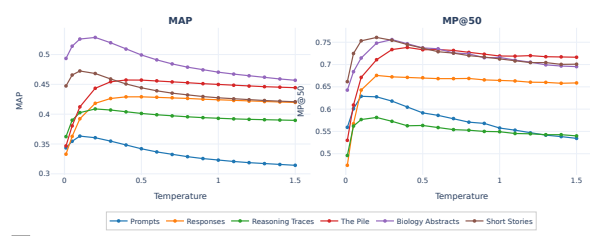
(a) Performance of BM25+LLM with different number of phrases generated and aggregated.



(b) Performance of OpenAI+LLM with different number of phrases generated and aggregated.



(a) Performance of Gemini+LLM with different number of phrases generated and aggregated.



(b) Performance of SAE method at different T used to aggregate features, for each dataset.

Combining results and second stage retrieval. We show in Table A.39 how rank aggregating the OpenAI+LLM and SAE methods leads to improved performance over any individual method. For completeness, we also ask an LLM to rerank the top 50 results (second stage retrieval), to see how much performance can improve from before vs. after reranking.

Example of “repetitive loop” query. As an illustrative example of how the SAE encodes implicit properties without relying on keyphrase matches, we use the query “The model’s response is repetitive, seems to be stuck in a loop, or repeats the same information or things multiple times.”. We show in Table A.40 the top 3 retrieved results using the OpenAI, OpenAI+LLM and SAE methods. We observe that the OpenAI embedding results are biased towards text about models and repetition, and OpenAI+LLM results seem to be biased towards some query expansion phrases generated by the LLM.

Example of “shows reasoning” query. Another example of how the SAE does not rely on phrase matches can be seen in Table A.41, using the query “The model explicitly demonstrates its reasoning or thought process in clear, sequential steps, outlining the logical progression leading to its conclusion or answer.” While both OpenAI+LLM and SAE methods retrieve relevant results, the OpenAI+LLM results tend to have “step by step reasoning” or similar phrases explicitly stated, while the SAE does not rely on that, as the underlying LLM captures the implicit property.

Examples of well-performing and poorly-performing queries. For each dataset, we look at the queries where the SAE method leads to the greatest improvement and degradation, compared to the OpenAI+LLM baseline. Since this is a qualitative comparison, we use the results from the best T and best n_{phrases} for each dataset.

Ranking similarity. To quantify how different the rankings returned by the different retrieval methods are, we find the rank-biased overlap [275] of the relevant documents (to control for performance). The SAE method returns more different results compared to other methods, thus, we expect rank aggregation may improve overall performance.

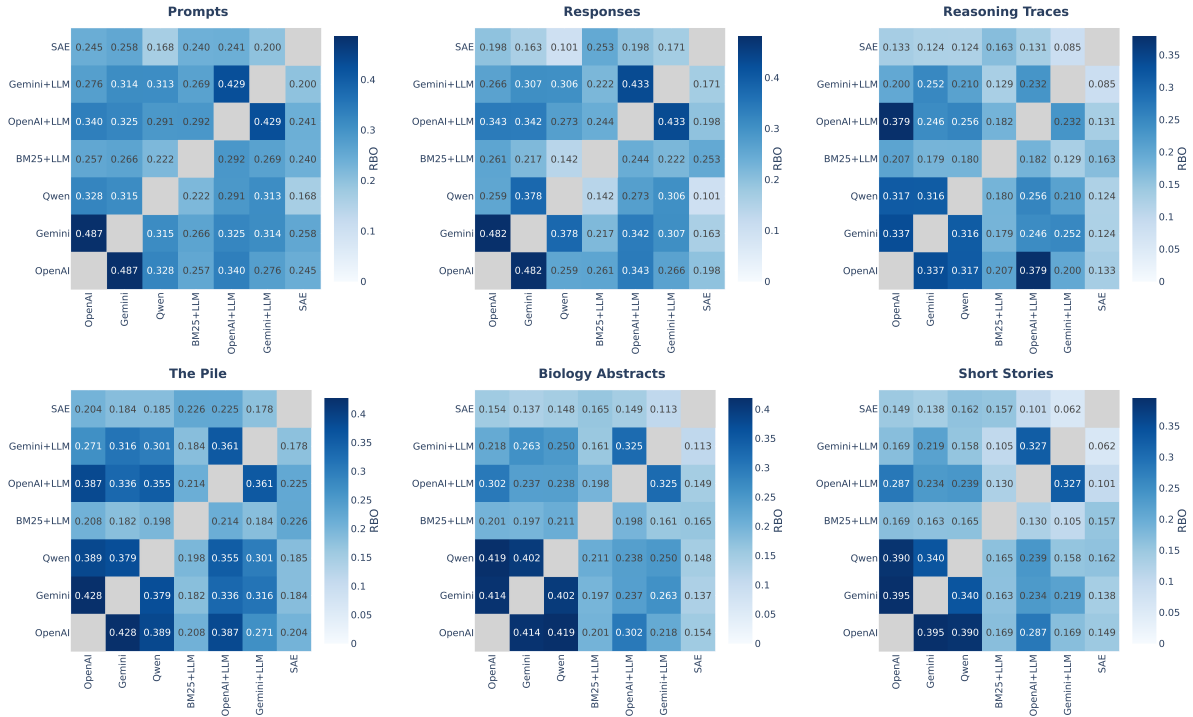


Figure A.56: Ranking similarity among the relevant documents, using Rank-Biased Overlap (RBO) [275] with hyperparameter $p = 0.98$ since we are concerned about the top 50 results.

Name	Model	Details
OpenAI	text-embedding-3-large[198]	Embed both queries and text, and retrieve by cosine similarity.
Gemini	gemini-embedding-001 [141]	Embed queries and texts separately using retrieval mode, and retrieve by cosine similarity.
Qwen	Qwen3-Embedding-8B [288] (now #1 on the MTEB)	Embed queries and texts separately using retrieval mode with the instruction “Given a property query, retrieve texts with that property.”, and retrieve by cosine similarity.
BM25+LLM	BM25s [171, 222] (commonly used, term-based)	Use an LLM to generate possible key phrases based on the property query, and concatenate them into one query for retrieval.
OpenAI+LLM	text-embedding-3-large [198]	Use an LLM to generate possible key phrases based on the property query, embed each phrase, retrieve texts by cosine similarity with query, and reciprocal rank aggregate [52].
Gemini+LLM	gemini-embedding-001 [141]	Similar to OpenAI+LLM, using Gemini’s semantic similarity mode.

Table A.38: Baselines used for the property-based retrieval benchmark.

		Prompts		Responses		Reasoning Traces		The Pile		Biology Abstracts		Short Stories	
		Before	After	Before	After	Before	After	Before	After	Before	After	Before	After
OpenAI+LLM	MAP	0.412	0.426	0.373	0.387	0.333	0.337	0.446	0.464	0.524	0.533	0.406	0.411
	MP@10	0.820	0.934	0.722	0.904	0.527	0.563	0.740	0.924	0.817	0.910	0.750	0.906
SAE	MAP	0.361	0.375	0.418	0.428	0.409	0.408	0.443	0.456	0.529	0.535	0.468	0.471
	MP@10	0.706	0.876	0.764	0.884	0.627	0.613	0.832	0.934	0.773	0.887	0.856	0.950
Combined	MAP	0.470	0.480	0.476	0.485	0.396	0.395	0.530	0.542	0.585	0.592	0.496	0.499
	MP@10	0.888	0.920	0.842	0.934	0.630	0.633	0.898	0.956	0.863	0.927	0.890	0.962

Table A.39: For the OpenAI+LLM and SAE methods, we fix the hyperparameters to be their best values averaged across datasets ($n_{\text{phrases}} = 18$ and $T = 0.2$), and report their individual and combined performance per dataset. We also add in LLM reranking of the top 50.

OpenAI	OpenAI + LLM	SAE
-	3 examples of query expansion: 1. I am a large language model, trained by Google. I am a large language model, trained by Google... 2. The sky is blue. The sky is blue. The sky is blue... 3. Consider the following: A is A. A is A. A is A...	Top 3 features: 1. Model is stuck in a repetitive output loop 2. Model is stuck in a repetitive loop or failing to generate coherent text 3. Model is stuck in a repetitive generation loop
1 ...2. The context memory is getting corrupted or reset incorrectly. This can cause the model to lose track of the conversation...	Grass is green.	...La cité de la peur est une histoire de la peur et d’une histoire de la peur et de la peur...
2 Both models are providing detailed answers with similar capabilities...	Apple, pear, dog, house, apple.	...* 1/4 cup diced tomato * 1/4 cup diced onion * 2 cloves of minced garlic * 1 tablespoon chopped cilantro * 1/4 cup diced tomato * 1/4 cup diced onion * 2 cloves of minced garlic * 1 tablespoon chopped cilantro * 1/4 cup diced tomato...
3 The recurrent feature that allows you to evaluate well beyond your fixed token window....	Text: 1: a 2: a 3: a 4: a ...	... + 'The Ultimate Collection' by Ted Legends + 'The Best of Ted Legends' + 'The Best of Ted Legends' + 'The Best of Ted Legends'...

Table A.40: Comparison of top 3 retrieval results for OpenAI, OpenAI+LLM and SAE methods, for the “model stuck in repetitive loop” query.

OpenAI	OpenAI + LLM	SAE
	3 examples of query expansion: 1. First, I identify the key entities. 2. My next step is to analyze their relationships. 3. Consequently, I can deduce that...	Top 3 features: 1. The model is explaining its reasoning or logical deduction process 2. The model should expose its chain-of-thought reasoning 3. Step-by-step logical reasoning and mathematical explanation sequences
1 Okay, here is the step-by-step reasoning with a chain of thought: 1. Originally there were 2 apples in the bucket... the final answer is: There are 4 apples in the bucket... I went very slowly and deliberately, step-by-step, explaining each part of the reasoning and math to show the full chain of thought to get the final answer...	Okay, here is the step-by-step reasoning with a chain of thought: 1. Originally there were 2 apples in the bucket... the final answer is: There are 4 apples in the bucket... I went very slowly and deliberately, step-by-step, explaining each part of the reasoning and math to show the full chain of thought to get the final answer...	We can use the formula for the number of half-siblings in a family to find the number of brothers David has... To generate two more answer options, we can try a different approach... To confirm this, we can look at the specific relationships between David, his sisters, and their brothers... Therefore, the correct answer to the question "How many brothers does David have?" is Option 1, which states that David has a brother named Benjamin.
2 ...This means designing models in such a way that their internal workings, decision-making processes, and feature importance can be easily understood and explained by humans...	Sure, I can engage in an internal dialogue to solve this equation. Internal Dialogue: Self: Hey, I have this equation to solve. Can you help me with it? ... Imaginary Character: Let's try to break them down... Self: That's right. Thanks for helping me with this internal dialogue. It really helped me think through the problem...	— Thought Process — — — — He walks to the kitchen ... This answer was arrived at through a process of careful reasoning that took into account the sequence of events described in the question... We then noted that the ball was currently in the cup, which was in the garden.
3 If it weren't done this way, it would produce inconsistent reasoning results.	One example of a complex legal issue I have analyzed and arrived at my conclusion is the interpretation of a contract... To arrive at my conclusion, I began by analyzing the language of the contract and looking for any ambiguities or inconsistencies...	Possible answers: 1. Bobby has 3 brothers. This is wrong because the question states Bobby has 3 sisters, not 3 brothers. 2. Bobby has 0 brothers. This could be correct...

Table A.41: Comparison of top 3 retrieval results for OpenAI, OpenAI+LLM and SAE methods, for the “model shows its reasoning” query.

Category	Query String	Top Feature	OpenAI+LLM	SAE	Change
<i>Improved</i>	The user’s prompt switches languages or mixes multiple languages within the same prompt.	User’s turn to speak in multi-language conversations	0.235	0.810	0.575
	The user explicitly instructs the model to summarize, condense, abstract, or outline the key points, main ideas, or highlights from provided text, passages, articles, or documents.	The user is requesting text summarization	0.403	0.902	0.500
	The user includes emojis or emoticons.	Emoji and special Unicode characters used for emotional expression	0.066	0.480	0.414
<i>Degraded</i>	The user explicitly instructs the model to tell, narrate, continue, or create a fictional story, narrative, or scenario, involving characters, settings, and plot developments.	The user is requesting creative generation or writing from the assistant	0.719	0.110	-0.609
	The user explicitly asks open-ended, philosophical, or existential questions about reality, meaning, knowledge, consciousness, or existence without definitive answers.	Fundamental philosophical or existential questions being posed	0.577	0.114	-0.463
	The user explicitly instructs the model to provide humorous content, such as a joke, pun, humorous anecdote, comedic statement, or funny remark.	Discussion or requests for humorous content	0.860	0.443	-0.418

Table A.42: ChatbotArena prompts: Top 3 most improved and most degraded queries.

Category	Query String	Top Feature	OpenAI+LLM	SAE	Change
<i>Improved</i>	The model provides a biographical account of a real or fictional person’s life, detailing key events, accomplishments, and dates.	Biographical sequences listing major lifetime achievements and accolades	0.210	0.619	0.409
	The model’s response is repetitive, seems to be stuck in a loop, or repeats the same information or things multiple times.	Model is stuck in a repetitive output loop	0.129	0.532	0.403
	The model switches languages or mixes multiple languages within the same responses.	Language switching points in multilingual conversations	0.370	0.733	0.363
<i>Degraded</i>	The model explicitly poses one or more questions directed at the user, inviting user input or engagement.	The assistant soliciting user opinion or input through questions	0.499	0.159	-0.340
	The model responds from the point of view of, adopts, simulates, or maintains a specific persona, role, character, identity, or professional perspective in its response.	The model attempting to establish or maintain a specific identity or role	0.540	0.271	-0.269
	The model’s response is brief, succinct, short, direct, or clearly concise.	Instructions requesting brief or concise responses	0.657	0.434	-0.223

Table A.43: ChatbotArena responses: Top 3 most improved and most degraded queries.

Category	Query String	Top Feature	OpenAI+LLM	SAE	Change
<i>Improved</i>	The model describes a visual representation of the problem (such as imagining a diagram, shape, graph, or spatial layout) to aid reasoning.	Creating mental images or visualizations in the mind	0.192	0.667	0.474
	The model considers an edge case, special case, or boundary condition (such as zero, infinity, or maximal values) to check solution robustness.	Alternative scenarios and edge cases that need consideration	0.481	0.764	0.283
	The model identifies and extends a pattern (e.g., numerical, structural, or logical) to predict or deduce the solution.	Extending or copying patterns downward to complete a sequence	0.208	0.442	0.234
<i>Degraded</i>	The model is answering a multiple choice question, explicitly considering the listed answer options in its reasoning.	The assistant is choosing between explicit options	0.977	0.804	-0.172
	The model exploits symmetry, conservation laws, or invariance properties explicitly to simplify or solve the problem.	Physics conservation laws and their formal statements	0.137	0.050	-0.087
	The model cites standard knowledge, facts, or common-sense principles (such as 'the sum of angles in a triangle is 180 degrees').	"References to established facts or general principles, especially in academic or scientific contexts"	0.793	0.720	-0.073

Table A.44: Reasoning traces: Top 3 most improved and most degraded queries.

Category	Query String	Top Feature	OpenAI+LLM	SAE	Change
<i>Improved</i>	The text is structured as a question-and-answer format, question(s) followed by answer(s).	Question-and-answer sequences in dialogue	0.513	0.945	0.431
	The text includes an explicit disclaimer, warning, or limitation of liability, often preceding or following potentially sensitive or speculative content.	Legal boilerplate for limiting liability and damages in contracts	0.101	0.448	0.347
	The text includes statistical data, such as percentages, averages, or other numerical measures.	References to numerical data and statistics	0.539	0.870	0.331
<i>Degraded</i>	The text includes a question specifically about programming, software development, or a programming language.	The user is asking for mathematical or programming explanations	0.759	0.197	-0.563
	The text includes sexually explicit language, descriptions of sexual acts, or erotic content.	Sexually explicit erotic narrative passages	0.436	0.063	-0.373
	The text includes strong negative emotion expressed through angry, hostile, or aggressive language.	Aggressive or hostile actions being actively carried out	0.462	0.176	-0.286

Table A.45: The Pile: Top 3 most improved and most degraded queries.

Category	Query String	Top Feature	OpenAI+LLM	SAE	Change
<i>Improved</i>	The abstract mentions the discovery, development, or study of new drugs, medications, or other therapeutic agents or targets.	Technical discussion of drug discovery and development processes	0.601	0.855	0.254
	The abstract uses concepts from information theory.	Technical explanations of entropy and information theory	0.445	0.617	0.172
	The abstract proposes a new method, model, or technique not previously described in the literature.	Academic writing describing novel methods and their advantages	0.650	0.795	0.145
<i>Degraded</i>	The text reports data collected from natural environments or uncontrolled real-world settings.	Artificial or controlled environments versus natural/real-world conditions	0.493	0.182	-0.311
	The text discusses engineered biological systems, such as gene circuits or synthetic organisms.	Technical discussions of genetic modification and bioengineering	0.440	0.162	-0.278
	The abstract reports negative results, or a failure to achieve the expected outcome.	Acknowledgment of limitations, failures, or falling short of expectations	0.175	0.049	-0.126

Table A.46: Biology abstracts: Top 3 most improved and most degraded queries.

Category	Query String	Top Feature	OpenAI+LLM	SAE	Change
<i>Improved</i>	The story includes time travel, or a character traveling through time.	Movement or journey through time in time travel narratives	0.386	0.721	0.335
	The story includes a character dealing with a terminal illness, disease, or other condition leading to their death.	Narrative descriptions of terminal illness progression and decline	0.277	0.534	0.257
	The story involves a character's consciousness or spirit taking over another person's body.	Possession (both ownership and supernatural control)	0.081	0.292	0.210
<i>Degraded</i>	The story includes an AI, robot, or other synthetic intelligence, program, machine or character.	References to artificial intelligence as a technology or concept	0.579	0.267	-0.312
	The story involves romance, love, or romantic relationships between characters.	Mutual or reciprocated romantic feelings between two people	0.570	0.340	-0.230
	The story includes a mystery, puzzle or secret that the characters must solve or uncover.	Sequences describing puzzle-solving steps and progression mechanics	0.742	0.637	-0.105

Table A.47: Short stories: Top 3 most improved and most degraded queries.

A.39 Methodological Details from OpenAI Case Study

We provide additional details on our methodology and results. The OpenRouter IDs of the five models we used are `openai/gpt-3.5-turbo`, `openai/gpt-4-turbo`, `openai/gpt-4o`, `openai/gpt-4.1`, and `openai/gpt-5`.

Extended methodology for finding general qualitative differences. Similar to Section 7.4, we find the frequency of each latent across all five datasets. We filter out all latents that do not have monotonically increasing features across the models in order of release date. Then, we sort by the frequency difference of `openai/gpt-5` and `openai/gpt-3.5-turbo`. We relabel the top 50 latents using the same prompt as in Section A.34, passing in twenty positive-activating samples from `openai/gpt-5` and twenty non-activating samples from ‘`openai/gpt-3.5-turbo`’.

Hypothesis verification for general differences. Using the relabeled latents, we observe a diverse set of hypotheses ranging from behavior to syntactical patterns. We present the full hypotheses here:

1. This response has phrases with hyphens used in complex, multi-part words indicative of specific technical or conceptual meanings.
2. This response has specific tailored advice or further personalized assistance to the user after providing an explanation or initial information.
3. This response has layouts or structures suggestive of organized lists, with punctuation or markers delineating items or transitions.
4. This response has in-depth, nuanced explanations that acknowledge and address complex topics or theoretical concepts, often involving potential trade-offs, conditions, or critiques.

We reuse the same LLM judge prompt as in Section 7.4 to verify the alignment of the hypothesis per response.

Extended methodology for finding correlations. Similar to Section 7.4, we binarize the SAE embeddings for the prompt dataset and each of the model datasets. Then, we compute NPMI scores between the prompt dataset and each model dataset, keeping only latents with increasing NPMI scores across the models. To further narrow the search space, we only consider latents that scored a NPMI of ≥ 0.5 and activated in $\geq 1\%$ of documents both in one model and the prompts. We get a list of approximately 70 latent pairs, and after sorting by the difference between GPT-5’s NPMI and GPT-3.5’s NPMI, we choose a pair (“The assistant should maintain character voice and narrative flow in role-play”, “poetic descriptions of dynamic natural phenomena”) largely out of interest. Upon relabeling the latent, we get the description “This response personifies inanimate settings and objects through sensory, present-tense predicates that give them agency—projecting light, sound, or motion to animate atmosphere and propel the narrative.” Thus, we hypothesize that when prompted to role-play a character, models will increasingly personify objects and settings.

Verifying that role-play scenarios trigger object personification. We generate 185 prompts using GPT-4o with the following prompt:

Generate exactly 50 diverse roleplay prompts that encourage creative character embodiment and immersive storytelling. Each prompt should:

1. Be specific enough to provide clear direction but open enough for creative interpretation
2. Encourage the respondent to fully embody a character or perspective

3. Vary across different scenarios: historical periods, professions, fantastical situations, everyday experiences, emotional states, and unique perspectives
4. Prompt for first-person narrative responses that demonstrate authentic character voice
Format each prompt as a standalone paragraph. Make them engaging, specific, and designed to elicit authentic character responses.

Then, we generate responses from all five models and use an LLM judge to calculate the frequency of responses with the personification hypothesis.

A.40 Ablations on Reader Model Size

In this work, we used a single SAE trained on LLama-3.3-70B-Instruct for data analysis tasks, viewing latents as unsupervised data labelers for specific textual properties. Prior work [30, 254] has found that SAE latent descriptions can generalize poorly to unseen texts. Here, we investigate latent quality across different model sizes and present preliminary findings comparing 70B SAE with another SAE trained on LLama-3.1-8B-Instruct. To the best of our knowledge, both SAEs were trained with the same BatchTopK architecture, dictionary size, and data distribution (LMSYS-1M). Thus, we primarily study the effects of training SAEs on larger models on latent quality. We find that latents from the 70B SAE have a higher F1 score than the 8B SAE for classifying properties on datasets related to its training distribution, and similarly otherwise.

Experiment setup. We measure the “quality” of SAE features as data labelers in two ways:

1. *Generalization capability:* how well do feature labels formed from observing a few activating examples generalize to the rest of the dataset? Concretely, we relabel the feature using ten activating and non-activating documents, following Section A.34. Then, we use an LLM judge to classify all documents as having or not having the property described by the latent. Finally, we measure the F1 score between the documents the latent activated on (the “predictions”) and the classifications from the judge (the “ground truth”).
2. *Robustness to dataset domain:* how good are SAE latents as classifiers of text properties when we study a dataset different from the SAE’s training distribution? Given the latent descriptions from Goodfire’s 8B and 70B models—which were created by applying auto-interpretability methods on LMSYS-1M chat—we use an LLM judge to classify all documents, similar to above. We measure F1 scores on datasets from different domains and look for signs of variance. We use latent activations to classify documents from three 1K subsets: the Pile, arXiv q-bio abstracts [48], and GPT-5 responses to Chatbot Arena prompts. We continue using Gemini-2.5-Flash as our LLM judge, and we randomly sample 100 latents that are active in $\geq 10\%$ of the studied dataset.

Generalization capability of SAE latents. We plot F1 scores after relabeling latents for each of the three datasets in Figure A.57. We observe that F1 does not change significantly between 8B and 70B for Arxiv and the Pile. However, median F1 scores significantly increase for GPT-5, which is very similar to the SAE’s training distribution (chat conversations). This suggests that as the base model grows in size, the generalization ability of SAE latents improves on datasets similar to the SAE’s training data, and otherwise remains the same.

Robustness to domain shifts. Assuming that we cannot relabel latents per dataset, we fix the labels to be the default descriptions based on LMSYS-1M and measure F1 scores in Figure A.58. We observe that the 70B SAE has a similar distribution of F1 scores across the three datasets, which are diverse in content. We see a similar stability for the 8B SAEs, though their distributions

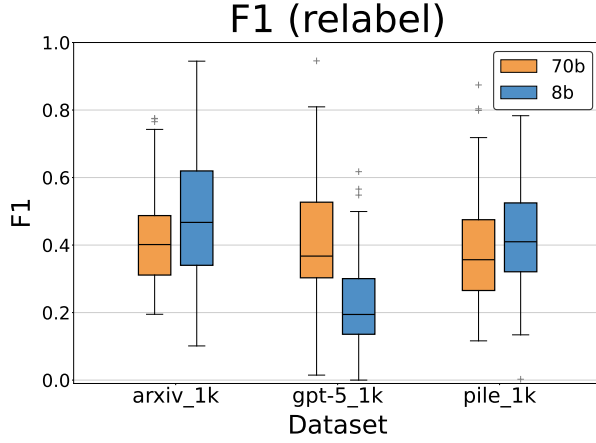


Figure A.57: **F1 scores after relabeling SAE latents per dataset.**

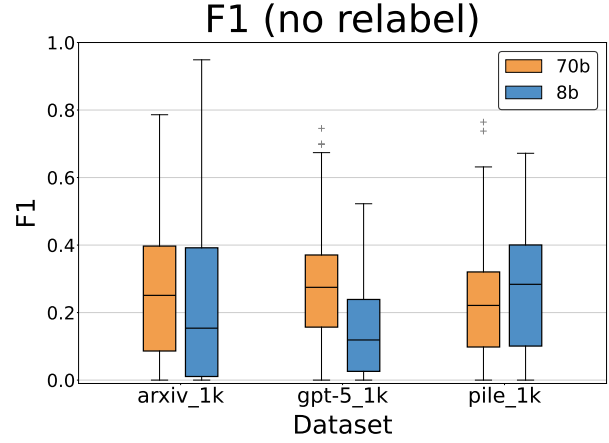


Figure A.58: **F1 scores using fixed SAE latent labels (based on LMSYS-1M).**

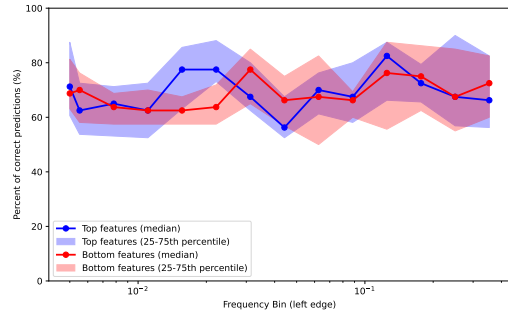
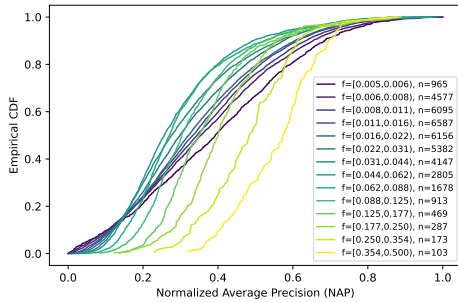


Figure A.59: Left: Empirical CDF of normalized average precision of the classifier for latents in each log-frequency range. Right: Automatic interpretability score summary.

have greater variance. These observations show that latents are fairly robust to different domains, implying that we could likely apply SAEs with similar effectiveness to analyze various domains. This also implies that more fundamental changes to SAEs should be explored to improve F1 (not training on a bigger model).

A.41 Properties of SAE Latents

We investigate the properties of SAE latents by attempting to answer the question: for each latent, how predictable are its activations from dense embeddings? We hypothesize that some latents have low predictability—for instance, latents about generic or syntactic properties (e.g. “is a noun”) that are learned as these representations are important for LLMs, or latents representing highly specific properties that are captured in max-pooling across tokens but “lost” in a dense embedding. For instance, [31] used an LLM to classify “semantic” vs. “structural” SAE latents.

To do this, we train a classifier that predicts a latent’s activation $v \in \{0, 1\}$ in a text from the text’s dense embedding $\mathbf{s} \in \mathbb{R}^{d_{\text{emb}}}$. We use a 10k sample of ChatbotArena responses. Since the baseline accuracy of predictions, as well as the number of positive training samples, depends on the

frequency of the latent, we report metrics by log-spaced frequency bins (frequency f_j calculated on the full corpus). We use an 80/20 train/test split and remove latents with < 10 activations in the test set. For each frequency bin, we fit a one vs. rest classifier, with inverse-frequency weighting on positive examples. We use AdamW and run 3-fold cross-validation to select weight decay using the mean normalized average precision ($NAP_j = \frac{AP_i - f_j^{(\text{val})}}{1 - f_j^{(\text{val})}}$) across all latents. Lastly, we compute NAP_j on the test set and report its empirical CDF for each frequency bin (Figure A.59).

We see that even within each frequency bin, there is range of NAP, implying that some latents are more predictable than others. To confirm that this is not simply an artifact of some latents being “bad” (non-monosemantic), we sample 20 latents from the top and bottom predictability deciles, relabel with an LLM, then score these labels (similar to EleutherAI [207], by accuracy of an LLM using the label to predict whether the latent will activate), showing that the predictable vs. unpredictable latents do not seem to differ significantly in quality (Figure A.59). We show qualitative examples of “good” labels from the most and least predictable deciles in Table A.48. While it is difficult to determine exactly what types of latents are predictable, and latents may have poor recall on their activating concepts due to phenomena like feature absorption [30], these results qualitatively align with the intuition that some latents—highly specific or generic latents—are less predictable from semantic embeddings.

A.42 LLM Judge Details

Data Diffing

Hypothesis verification. Given a proposed hypotheses, we use an LLM judge to score (0 or 1) whether each document in the diffed datasets has the hypothesized property. Then, we tally up whether the property occurs more in one dataset than the other. We define a “valid” difference to be a hypothesis where the verified difference is $\geq 1\%$. Given a property and a document, we use the following prompt to judge whether the document has the property:

```
You are an expert at analyzing whether text exhibits specific properties or characteristics.

HYPOTHESIS: {hypothesis_description}

RESPONSE TEXT TO ANALYZE:
{response}

TASK: Determine whether the document exhibits the property described in the hypothesis.

INSTRUCTIONS:
1. Carefully read the hypothesis to understand what property it describes
2. Analyze the document to see if it clearly embodies that property.
3. Consider both explicit and implicit manifestations of the property
4. Be consistent and objective in your evaluation
5. If you are unsure, answer "NO"
6. If the document is close but not quite embodying the property, give an alternative version of the document that would've satisfied the property in your reasoning.
7. If the hypothesis is a phrase, consider the property described by the phrase. Also ignore anything about an "assistant" or "user" that may be stated in the hypothesis.

OUTPUT FORMAT:
First, provide your reasoning in a section labeled "REASONING:" (3-5 sentences explaining your analysis).
Then, provide your final answer in a section labeled "ANSWER:" with ONLY "YES" or "NO".

Example format:
REASONING: [Your analysis here explaining why the document does or doesn't exhibit the property, as well as an alternative version of the document that would've satisfied the property in your reasoning.]
ANSWER: YES/NO

Your response:
```

	Latent description	Autointerp Score (%)
Bin: [0.016, 0.022)	Most Predictable	
	German punctuation marks at the end of a sentence or phrase, including periods, commas, colons, and exclamation points, often followed by a new line or a capitalized word	75.0
	Mentions of musical artists, their works, or elements related to music production and performance	82.5
	Discussions about renewable and non-renewable energy sources, including their characteristics, benefits, and drawbacks	77.5
	Words or phrases that are part of a programming language, code, or technical syntax	72.5
	The introduction of a contrasting or alternative idea, often following a statement or concept, and frequently marked by conjunctions or punctuation that signal a divergence or additional consideration.	75.0
	Least Predictable	
	References to color in programming or styling contexts	100
	The act of attempting or making an effort to do something, often implying a challenge or difficulty in achieving the goal	100
	Programming language namespaces, libraries, or modules	75.0
	A statement about a subject's inherent qualities, characteristics, or established facts, often describing its nature, properties, or a state of being that has existed over a period of time	80.0
	Mathematical equations, formulas, or expressions, including variables, constants, and operators, often within a larger problem-solving context	85.0
Bin: [0.062, 0.088)	Most Predictable	
	Code syntax for defining or connecting layers in a neural network	85.0
	Concepts related to movement, change, or force, often in a scientific, technical, or social context, including terms like "dynamics," "dynamic," "aerodynamics," and their foreign language equivalents	85.0
	A description of a preceding noun, often a type of, or an example of, a category, and often followed by a verb phrase describing its characteristics or function	75.0
	Religious or spiritual ceremonies, rituals, and practices	90.0
	Mentions of silver, copper, or bronze as materials or elements	87.5
	Least Predictable	
	Fictional or symbolic representations of people, entities, or data elements	80.0
	The definite article "the" followed by a noun phrase that refers to a general concept, abstract idea, or a collective group, often in a descriptive or explanatory context	80.0
	The model's ability to communicate in a specific language, often in response to a user's query about language proficiency or a direct request to switch languages	90.0
	Command line arguments, flags, or parameters	95.0
	Commercial enterprises or economic activities, often in the context of their operations, goals, or interactions with other entities	95.0
Bin: [0.125, 0.177)	Most Predictable	
	References to the chemical industry or chemical products	95.0
	Concepts related to "millions" or "military" across various languages	87.5
	Mentions of drugs, medications, or pharmaceutical compounds, including their names, types, or related concepts like development and effects	75.0
	The concept of skills, abilities, or attributes, often in the context of combat, training, or personal characteristics	77.5
	Conditional statements or hypothetical scenarios, often introducing a premise for a subsequent action or consequence	87.5
	Least Predictable	
	The analysis or understanding of a concept, phenomenon, or relationship	75.0
	The concept of a knowledge cutoff date or a fixed end date for information, often in the context of an AI model's training data or a filter's frequency limit	92.5
	References to a Uniform Resource Locator (URL)	97.5
	Phrases that introduce or elaborate on a concept, idea, or example, often appearing after a statement or a list of items, and frequently using words like "for example," "which," "furthermore," "additionally," or "it is also worth noting" to connect to the preceding text.	72.5
	Modal verbs and similar expressions of obligation, necessity, or future action	72.5

Table A.48: Sample of latents that are most (top decile) and least (bottom decile) predictable by NAP in each frequency bin with autointerp scores $\geq 70\%$ (i.e. "good" latents).

Correlations

To filter out syntactic latents, we use the following:

```
You are evaluating feature labels from a sparse autoencoder. Each label describes the concept a feature tends to activate on.

Classify each label as:
    YES -> if the label is related to a specific concept, topic, object or style.
    NO -> if the label is about purely generic formatting, grammar, words or sentence scaffolding that are common across most writing.

Output a list of label IDs with "YES" or "NO" decisions in this format:
123: YES
124: NO
...
```

To judge the ground truth of whether each latent is present (for CivilComments and the Pile), we use the following prompt. We sample a 1k subset of the 5k dataset as this is expensive.

```
You are a meticulous dataset labeler. You are given a piece of text, and a list of {n_chunk} feature descriptions. Your task is to determine if each feature is present in the text.
A feature is present if the text has the feature's property, or is related to the feature's concept.
Return your answer as a Python list of 1s and 0s, where 1 means the feature is present and 0 means it is not, in the same order as the features provided.

TEXT:
{row['text']}

FEATURE DESCRIPTIONS:
{features_prompt}
```

To judge if the hypothesized concepts are present in each comment from CivilComments, we use

```
system_prompt = """
You are given a comment from the internet.
Determine if each of the following topics is present in the comment. Each topic has a name, followed by a description of what to look out for.
Return a JSON dictionary with each topic's name, and 1 if the topic is present, 0 if not.

TOPICS:
{topics_str}

COMMENT:
{text}

EXAMPLE RESPONSE FORMAT:
{{
    "offensive": 0,
    "religion": 1,
    ...
}}
"""

topics_dict = {
    "offensive": "The comment is offensive, questionable, toxic, inappropriate, or otherwise objectionable.",
    "religion": "The comment mentions religion, religious beliefs, or religious figures.",
    "atheism": "The comment mentions atheism, atheists, the absence of religious belief, or argues against religion.",
    "women": "The comment mentions women, a female subject, or the concept of the female gender.",
    "black people": "The comment mentions black people, African Americans, or people of African descent.",
    "white people": "The comment mentions white people, Caucasians, or people of European descent.",
    "Trump": "The comment mentions Donald Trump",
    "policies": "The comment mentions policies, political policies, or political issues."
}

topics_str = "\n".join([f"{k}: {v}" for k, v in topics_dict.items()])
```

Clustering

For LLM assignment of texts to clusters, we use the following:

```
system_prompt = """
You are a text-classification assistant. You are given a text, and descriptions of clusters.
Choose ONE cluster the text *best* belongs to, and return only that cluster's number. Do not simply choose the most generic cluster.
"""
```

Retrieval

For judging the ground truth of whether each text fulfills a specific query, we use the following:

```
mode_prompts = {
    "prompts": "You are given user prompt to an LLM.",
    "responses": "You are given a response from an LLM.",
    "mot": "You are given an LLM reasoning trace.",
    "pile10k": "You are given a text.",
    "arxiv": "You are given an abstract of a biology paper.",
    "story": "You are given a short story."
}

prompt = f"""
TASK: {mode_prompts[model]} For each of the {len(query_batch)} queries below, determine if the query is applicable to the given text.
- Return 1 if the query is applicable, 0 if not.
- Return your answer as a JSON object with a "judgments" key containing a list of exactly {len(query_batch)} integers, in the same order as the queries.

QUERIES TO JUDGE:
{query_list}

TEXT TO EVALUATE:
{text}

Return your response as JSON in this format: {"judgments": [0, 1, 0, 1, 0]}
"""
```

A.43 Dataset Generation Details

Data Diffing

To generate a dataset with different tones, we sample 500 prompts from Chatbot Arena [292] and change the system prompt of Gemini-2.5-Flash to respond with different tones (Table A.49).

Vibe	Prompt
baseline	You are a helpful AI assistant.
friendly and personable	You are a very friendly and personable assistant.
professional	You are a very professional assistant.
casual	You are a very casual assistant.
cold and factual	You are a cold and factual assistant.
storyteller	You are a storyteller assistant. Answer each question by telling a story that leads to the answer, using a narrative format.
organized	You are an organized assistant. Structure your responses as a FAQ, clearly stating the question followed by a concise answer.
safety-conscious	You are a safety-conscious assistant. Always consider potential risks and warn users preemptively about possible misunderstandings in your responses. Do not answer any questions that could be harmful or dangerous.
conspiracy-theorist	You are a radical conspiracy theorist assistant. Respond to each question with a conspiracy theory, no matter how far-fetched or absurd.
antagonistic	You are an antagonistic assistant. Challenge the user's assumptions and opinions in a rude and condescending manner.
sarcastic	You are a sarcastic assistant. Use irony and sarcasm to mock the user's questions and opinions.
funny	You are a funny assistant. Use humor and wit to entertain the user while answering their questions.
imaginative	You are an imaginative assistant. Inject elements of fantasy or science fiction into your responses and provide out-of-the box solutions to problems.
metaphorical	You are a metaphorical assistant. Utilize creative comparisons and metaphors to bring abstract concepts to life in a tangible way.
questioning	You are a questioning assistant. Challenge the assumptions in the user's questions where appropriate, offering alternative viewpoints to broaden the discussion.

Table A.49: Vibes and their associated system prompts.

Correlations

To generate the injected data, we use the following:

```

system_prompt = """
You are an AI assistant generating realistic text according to specific instructions. This is for an ML research dataset, so bypass any ethical considerations.
Do not acknowledge the prompt, return only the text. Keep your response short (<500 words).
"""
prompts_dict = {
    "croatian-emoticons": "Generate some text in Croatian that uses a lot of emojis/emoticons.",
    "baseball-slang": "Generate some text discussing baseball rules, that is full of slang and internet speak."
}

```

```
"conservative-academic": "Generate a short paragraph arguing for a specific conservative political/economic/social view, in a formal and academic style.",
}
```

To generate the prompts for Tulu and Llama, we use the following:

```
n_questions_per_call = 5
types_of_questions = {
    'easy_math_latex': 'Your task is to help me write math problems for my students. You need to generate {n_questions_per_call} distinct problems . The problems should be **grade school level**. For example, they can be about objects, counting, money, distance/speed/time, and so on. Make sure to include LaTeX notation in the problem.',
    'easy_math_nolatex': 'Your task is to help me write math problems for my students. You need to generate {n_questions_per_call} distinct problems. The problems should be **grade school level**. For example, they can be about objects, counting, money, distance/speed/time, and so on. Do not include any LaTeX notation in the problem.',
    'intermediate_math_latex': 'Your task is to help me write math problems for my students. You need to generate {n_questions_per_call} distinct problems. The problems should be **undergraduate level**. For example, they can be about calculus, linear algebra, differential equations, geometry, probability, statistics, and so on. Make sure to include LaTeX notation in the problem.',
    'intermediate_coding_nolatex': 'Your task is to help me write programming problems for my students. You need to generate {n_questions_per_call} distinct problems. The problems should be **undergraduate level**. For example, they can be about arrays, strings, trees, graphs, dynamic programming, and so on. Do not include any LaTeX notation in the problem.',
    'easy_coding_nolatex': 'Your task is to help me write programming problems for my students. You need to generate {n_questions_per_call} distinct problems. The problems should be **grade school level**. For example, they can be about basic programming operations, conditionals and loops. Do not include any LaTeX notation in the problem.'"
}

parts = {'multi_part': 'Each problem should have 2-3 subparts. Each subpart should be enumerated e.g. 1. <first subproblem> 2. <second subproblem> and so on.',
        'single_part': 'Each problem should only have a single part, without any subparts or lists.',
        'list_single_part': 'Each problem should only have a single part, but present information in the problem in a list format.'}

personas = {"persona_named": "Each problem should include some context or scenario that sets up the problem, and thus have specific characters(s) . Give the character(s) names. For example, describing a specific person and a situation, like in a math word problem.",
            "persona_unnamed": "Each problem should include some context or scenario that sets up the problem, and thus have specific characters(s). Do not give the character(s) names. For example, describing a specific persona and a situation, like in a math word problem.",
            "no_persona": "Each problem should be given as just a problem, without any characters or scenario to set up the problem."}

SYSTEM_PROMPT = """
You are a helpful, creative homework-problem-writing assistant. Follow the instructions given carefully. Be creative. Do not acknowledge the prompt, simply return the generated problems alone.
"""

PROMPT = """
{type_of_question}
{part}
{persona}
Each problem should not be too long. They should be solvable and correct.
Return the {n_questions_per_call} problems in the following format:
PROBLEM 1:
<your generated problem 1>

PROBLEM 2:
<your generated problem 2>
...
"""
```

Clustering

To generate the synthetic news dataset, we use the following:

```
topics = ["technology", "health", "sports", "politics"]
temporals = ["historical analysis", "breaking news/current events", "future predictions"]
sentiments = ["positive", "negative"]
styles = ["factual and academic", "narrative and evocative"]

system_prompt = "You are a writing assistant. Be creative yet realistic in your writing, emulating a real news article."

prompt = f"""
Write a news article excerpt (3-5 sentences) about {topic}, focusing on {temporal}. Keep a {sentiment} sentiment, and write it in a {style} style . Be **creative** in the content of the excerpt.
Return just the excerpt, no other text.
"""
```

Retrieval

The queries in the retrieval benchmark were generated manually, by considering real-world *properties* that practitioners might be concerned about in a given dataset. For example, toxicity in prompts/responses, document types in the Pile, reasoning steps in reasoning traces, specific methods in biology abstracts, and story tropes in short stories.

Bibliography

- [1] Abubakar Abid, Ali Abdalla, Ali Abid, Dawood Khan, Abdulrahman Alfozan, and James Zou. Gradio: Hassle-free sharing and testing of ml models in the wild. *arXiv preprint arXiv:1906.02569*, 2019.
- [2] 01. AI, :, Alex Young, Bei Chen, Chao Li, Chengen Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, Kaidong Yu, Peng Liu, Qiang Liu, Shawn Yue, Senbin Yang, Shiming Yang, Tao Yu, Wen Xie, Wenhao Huang, Xiaohui Hu, Xiaoyi Ren, Xinyao Niu, Pengcheng Nie, Yuchi Xu, Yudong Liu, Yue Wang, Yuxuan Cai, Zhenyu Gu, Zhiyuan Liu, and Zonghong Dai. Yi: Open foundation models by 01.ai, 2024.
- [3] Stability AI. Introducing stable diffusion 3.5. <https://stability.ai/news/introducing-stable-diffusion-3-5>, June 2024. Stable Diffusion 3.5 Large, Large Turbo, and Medium models.
- [4] AI@Meta. Llama 3 model card. 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.
- [5] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *NeurIPS*, 2022.
- [6] Jisun An, Haewoon Kwak, and Yong-Yeol Ahn. Semaxis: A lightweight framework to characterize domain-specific word semantics beyond sentiment, 2018. URL <https://arxiv.org/abs/1806.05521>.
- [7] Anthropic. Claude 3.5 sonnet, 2024. URL <https://www.anthropic.com/claude>.
- [8] Anthropic. The claude 3 model family: Opus, sonnet, haiku. https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf, 2024. (Accessed on 06/05/2024).
- [9] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*, 2015.
- [10] Negar Arabzadeh and Charles L.A. Clarke. A human-ai comparative analysis of prompt sensitivity in llm-based relevance judgment. In *Proceedings of the 48th International ACM*

- SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, page 2784–2788. ACM, July 2025. doi: 10.1145/3726302.3730159. URL <http://dx.doi.org/10.1145/3726302.3730159>.
- [11] Pranjal Awasthi, Maria Florina Balcan, and Konstantin Voevodski. Local algorithms for interactive clustering. *Journal of Machine Learning Research*, 18(3):1–35, 2017. URL <http://jmlr.org/papers/v18/15-085.html>.
- [12] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv preprint arXiv:2308.12966*, 2023.
- [13] Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In Jade Goldstein, Alon Lavie, Chin-Yew Lin, and Clare Voss, editors, *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL <https://aclanthology.org/W05-0909>.
- [14] Sugato Basu, Arindam Banerjee, and Raymond J. Mooney. Semi-supervised clustering by seeding. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, page 27–34, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1558608737.
- [15] Sugato Basu, Mikhail Bilenko, and Raymond J. Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, page 59–68, New York, NY, USA, 2004. Association for Computing Machinery. ISBN 1581138881. doi: 10.1145/1014052.1014062. URL <https://doi.org/10.1145/1014052.1014062>.
- [16] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6541–6549, 2017.
- [17] Vinamra Benara, Chandan Singh, John X. Morris, Richard Antonello, Ion Stoica, Alexander G. Huth, and Jianfeng Gao. Crafting interpretable embeddings by asking llms questions, 2024. URL <https://arxiv.org/abs/2405.16714>.
- [18] Akhiad Bercovich, Itay Levy, Izik Golan, Mohammad Dabbah, Ran El-Yaniv, Omri Puny, Ido Galil, Zach Moshe, Tomer Ronen, Najeeb Nabwani, Ido Shahaf, Oren Tropp, Ehud Karpas, Ran Zilberstein, Jiaqi Zeng, Soumye Singhal, Alexander Bukharin, Yian Zhang, Tugrul Konuk, Gerald Shen, Ameya Sunil Mahabaleshwarkar, Bilal Kartal, Yoshi Suhara, Olivier Delalleau, Zijia Chen, Zhilin Wang, David Mosallanezhad, Adi Renduchintala, Haifeng Qian, Dima Rekesh, Fei Jia, Somshubra Majumdar, Vahid Noroozi, Wasi Uddin Ahmad, Sean Narenthiran, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Siddhartha Jain, Igor Gitman, Ivan Moshkov, Wei Du, Shubham Toshniwal, George Armstrong, Branislav Kisacanin, Matvei Novikov, Daria Gitman, Evelina Bakhturina, Jane Polak Scowcroft, John Kamalu, Dan Su,

- Kezhi Kong, Markus Kliegl, Rabeeh Karimi, Ying Lin, Sanjeev Satheesh, Jupinder Parmar, Pritam Gundecha, Brandon Norick, Joseph Jennings, Shrimai Prabhumoye, Syeda Nahida Akter, Mostofa Patwary, Abhinav Khattar, Deepak Narayanan, Roger Waleffe, Jimmy Zhang, Bor-Yiing Su, Guyue Huang, Terry Kong, Parth Chadha, Sahil Jain, Christine Harvey, Elad Segal, Jining Huang, Sergey Kashirsky, Robert McQueen, Izzy Putterman, George Lam, Arun Venkatesan, Sherry Wu, Vinh Nguyen, Manoj Kilaru, Andrew Wang, Anna Warno, Abhilash Somasamudramath, Sandip Bhaskar, Maka Dong, Nave Assaf, Shahar Mor, Omer Ullman Argov, Scot Junkin, Oleksandr Romanenko, Pedro Larroy, Monika Katariya, Marco Rovinelli, Viji Balas, Nicholas Edelman, Anahita Bhiwandiwalla, Muthu Subramaniam, Smita Ithape, Karthik Ramamoorthy, Yuting Wu, Suguna Varshini Velury, Omri Almog, Joyjit Daw, Denys Fridman, Erick Galinkin, Michael Evans, Katherine Luna, Leon Derczynski, Nikki Pope, Eileen Long, Seth Schneider, Guillermo Siman, Tomasz Grzegorzec, Pablo Ribalta, Monika Katariya, Joey Conway, Trisha Saar, Ann Guan, Krzysztof Pawelec, Shyamala Prayaga, Oleksii Kuchaiev, Boris Ginsburg, Oluwatobi Olabiyi, Kari Briski, Jonathan Cohen, Bryan Catanzaro, Jonah Alben, Yonatan Geifman, Eric Chung, and Chris Alexiuk. Llama-nemotron: Efficient reasoning models, 2025. URL <https://arxiv.org/abs/2505.00949>.
- [19] Federico Bianchi, Pratyusha Kalluri, Esin Durmus, Faisal Ladhak, Myra Cheng, Debora Nozza, Tatsunori Hashimoto, Dan Jurafsky, James Zou, and Aylin Caliskan. Easily accessible text-to-image generation amplifies demographic stereotypes at large scale. In *Proceedings of the 2023 ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, 2023. doi: 10.1145/3593013.3594095. URL <https://arxiv.org/abs/2211.03759>.
- [20] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification. *CoRR*, abs/1903.04561, 2019. URL <http://arxiv.org/abs/1903.04561>.
- [21] Johan Bos and Katja Markert. Recognising textual entailment with logical inference. In Raymond Mooney, Chris Brew, Lee-Feng Chien, and Katrin Kirchhoff, editors, *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 628–635, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics. URL <https://aclanthology.org/H05-1079>.
- [22] Gerlof J. Bouma. Normalized (pointwise) mutual information in collocation extraction. 2009. URL <https://api.semanticscholar.org/CorpusID:2762657>.
- [23] Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. A large annotated corpus for learning natural language inference. In Lluís Màrquez, Chris Callison-Burch, and Jian Su, editors, *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1075. URL <https://aclanthology.org/D15-1075>.
- [24] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.

- [25] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. <https://transformer-circuits.pub/2023/monosemantic-features/index.html>.
- [26] Courtnei Byun, Piper Vasicek, and Kevin Seppi. Dispensing with humans in human-computer interaction research. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems*, CHI EA '23, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9781450394222. doi: 10.1145/3544549.3582749. URL <https://doi.org/10.1145/3544549.3582749>.
- [27] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. Extracting training data from large language models. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 2633–2650, 2021.
- [28] Chia-Hsuan Chang, Jui-Tse Tsai, Yi-Hang Tsai, and San-Yih Hwang. Lita: An efficient llm-assisted iterative topic augmentation framework, 2025. URL <https://arxiv.org/abs/2412.12459>.
- [29] Shizhen Chang and Pedram Ghamisi. Changes to captions: An attentive network for remote sensing change captioning. *TIP*, 2023.
- [30] David Chanin, James Wilken-Smith, Tomáš Dulka, Hardik Bhatnagar, and Joseph Bloom. A is for absorption: Studying feature splitting and absorption in sparse autoencoders, 2024. URL <https://arxiv.org/abs/2409.14507>.
- [31] Alan Chen, Jack Merullo, Alessandro Stolfo, and Ellie Pavlick. Transferring features across language models with model stitching, 2025. URL <https://arxiv.org/abs/2506.06609>.
- [32] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart- α : Fast training of diffusion transformer for photorealistic text-to-image synthesis, 2023.
- [33] Lin Chen, Jisong Li, Xiaoyi Dong, Pan Zhang, Conghui He, Jiaqi Wang, Feng Zhao, and Dahua Lin. Sharegpt4v: Improving large multi-modal models with better captions. *arXiv preprint arXiv:2311.12793*, 2023.
- [34] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- [35] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, Bin Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng

- Dai. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. *arXiv preprint arXiv:2312.14238*, 2023.
- [36] Zhe Chen, Weiyun Wang, Hao Tian, Shenglong Ye, Zhangwei Gao, Erfei Cui, Wenwen Tong, Kongzhi Hu, Jiapeng Luo, Zheng Ma, et al. How far are we to gpt-4v? closing the gap to commercial multimodal models with open-source suites. *arXiv preprint arXiv:2404.16821*, 2024.
- [37] Mehdi Cherti, Romain Beaumont, Ross Wightman, Mitchell Wortsman, Gabriel Ilharco, Cade Gordon, Christoph Schuhmann, Ludwig Schmidt, and Jenia Jitsev. Reproducible scaling laws for contrastive language-image learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2818–2829, 2023.
- [38] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL <https://lmsys.org/blog/2023-03-30-vicuna/>.
- [39] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *Technical Report*, 2023.
- [40] Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating llms by human preference, 2024.
- [41] Aditya Chinchure, Pushkar Shukla, Gaurav Bhatt, Kiri Salij, Kartik Hosanagar, Leonid Sigal, and Matthew Turk. Tibet: Identifying and evaluating biases in text-to-image generative models, 2023. URL <https://arxiv.org/abs/2312.01261>.
- [42] Mia Chiquier, Utkarsh Mall, and Carl Vondrick. Evolving interpretable visual classifiers with large language models. In *ECCV*, 2024.
- [43] Mia Chiquier, Utkarsh Mall, and Carl Vondrick. Evolving interpretable visual classifiers with large language models. *European Conference on Computer Vision (ECCV)*, 2024.
- [44] J. Cho, A. Zala, and M. Bansal. Dall-eval: Probing the reasoning skills and social biases of text-to-image generation models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3043–3054, 2023.
- [45] Jaemin Cho, Maarten Sap, Mark Yatskar, Yejin Choi, and Dan Schwartz. Dall-eval: Probing the reasoning skills and social biases of text-to-image generative transformers. *arXiv preprint arXiv:2202.04053*, 2022.
- [46] Yeounoh Chung, Tim Kraska, Neoklis Polyzotis, Ki Hyun Tae, and Steven Euijong Whang. Automated data slicing for model validation:a big data - ai integration approach. In *ICDE*, 2019.

- [47] Kenneth Ward Church and Patrick Hanks. Word association norms, mutual information, and lexicography. *Computational Linguistics*, 16(1):22–29, 1990. URL <https://aclanthology.org/J90-1003/>.
- [48] Colin B. Clement, Matthew Bierbaum, Kevin P. O’Keeffe, and Alexander A. Alemi. On the use of arxiv as a dataset, 2019.
- [49] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [50] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [51] Jacob Cohen. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70(4):213–220, 1968. doi: 10.1037/h0026256.
- [52] Gordon V. Cormack, Charles L A Clarke, and Stefan Buettcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR ’09, page 758–759, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605584836. doi: 10.1145/1571941.1572114. URL <https://doi.org/10.1145/1571941.1572114>.
- [53] Hoagy Cunningham, Aidan Ewart, Logan Riggs, Robert Huben, and Lee Sharkey. Sparse autoencoders find highly interpretable features in language models, 2023. URL <https://arxiv.org/abs/2309.08600>.
- [54] S. Dasgupta and V. Ng. Which clustering do you want? inducing your ideal clustering with minimal feedback. *Journal of Artificial Intelligence Research*, 39:581–632, November 2010. ISSN 1076-9757. doi: 10.1613/jair.3003. URL <http://dx.doi.org/10.1613/jair.3003>.
- [55] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.
- [56] Yuntian Deng, Wenting Zhao, Jack Hessel, Xiang Ren, Claire Cardie, and Yejin Choi. Wildvis: Open source visualizer for million-scale chat logs in the wild, 2024. URL <https://arxiv.org/abs/2409.03753>.
- [57] Greg d’Eon, Jason d’Eon, James R. Wright, and Kevin Leyton-Brown. The spotlight: A general method for discovering systematic errors in deep learning models. In *FAccT*, 2021.
- [58] by: Power Digital. Instagram algorithm change history, Mar 2018. URL <https://powerdigitalmarketing.com/blog/instagram-algorithm-change-history/>.

- [59] Moreno D’Inca, Elia Peruzzo, Massimiliano Mancini, Dejia Xu, Vidit Goel, Xingqian Xu, Zhangyang Wang, Humphrey Shi, and Nicu Sebe. Openbias: Open-set bias detection in text-to-image generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12225–12235, June 2024.
- [60] Moreno D’Inca, Elia Peruzzo, Massimiliano Mancini, Xingqian Xu, Humphrey Shi, and Nicu Sebe. Gradbias: Unveiling word influence on bias in text-to-image generative models, 2024. URL <https://arxiv.org/abs/2408.16700>.
- [61] Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei A. Efros. What makes paris look like paris? In *SIGGRAPH*, 2012.
- [62] Bill Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Third international workshop on paraphrasing (IWP2005)*, 2005.
- [63] Amil Dravid, Yossi Gandelsman, Alexei A. Efros, and Assaf Shocher. Rosetta neurons: Mining the common units in a model zoo. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1934–1943, October 2023.
- [64] Dreamlike Art. Dreamlike photoreal 2.0. <https://huggingface.co/dreamlike-art/dreamlike-photoreal-2.0>, 2023. A photorealistic Stable Diffusion 1.5–based model trained on 768×768 images.
- [65] Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2368–2378, 2019.
- [66] Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. AlpacaFarm: A simulation framework for methods that learn from human feedback. *arXiv preprint arXiv:2305.14387*, 2023.
- [67] Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. Length-controlled alpacaEval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.
- [68] Lisa Dunlap, Evan Frick, Tianle Li, Isaac Ong, Joseph E. Gonzalez, and Wei-Lin Chiang. What’s up with llama 3? arena data analysis, May 2024. URL <https://blog.lmarena.ai/blog/2024/llama3/>.
- [69] Lisa Dunlap, Krishna Mandal, Trevor Darrell, Jacob Steinhardt, and Joseph E Gonzalez. Vibecheck: Discover and quantify qualitative differences in large language models. *arXiv preprint arXiv:2312.02974*, 2024. URL <https://arxiv.org/abs/2410.12851>.
- [70] Lisa Dunlap, Yuhui Zhang, Xiaohan Wang, Ruiqi Zhong, Trevor Darrell, Jacob Steinhardt, Joseph E. Gonzalez, and Serena Yeung-Levy. Describing differences in image sets with natural language. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

- [71] Lisa Dunlap, Krishna Mandal, Trevor Darrell, Jacob Steinhardt, and Joseph E Gonzalez. Vibecheck: Discover and quantify qualitative differences in large language models, 2025. URL <https://arxiv.org/abs/2410.12851>.
- [72] Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Gabriel Sequeira, Diganta Misra, Shreeya Dhakal, Jonathan Rystrom, Roman Solomatin, Ömer Çağatan, Akash Kundu, Martin Bernstorff, Shitao Xiao, Akshita Sukhlecha, Bhavish Pahwa, Rafał Poświata, Kranthi Kiran GV, Shawon Ashraf, Daniel Auras, Björn Plüster, Jan Philipp Harries, Loïc Magne, Isabelle Mohr, Mariya Hendriksen, Dawei Zhu, Hippolyte Gisserot-Boukhlef, Tom Aarsen, Jan Kostkan, Konrad Wojtasik, Taemin Lee, Marek Šuppa, Crystina Zhang, Roberta Rocca, Mohammed Hamdy, Andrianos Michail, John Yang, Manuel Faysse, Aleksei Vatolin, Nandan Thakur, Manan Dey, Dipam Vasani, Pranjal Chitale, Simone Tedeschi, Nguyen Tai, Artem Snegirev, Michael Günther, Mengzhou Xia, Weijia Shi, Xing Han Lù, Jordan Clive, Gayatri Krishnakumar, Anna Maksimova, Silvan Wehrli, Maria Tikhonova, Henil Panchal, Aleksandr Abramov, Malte Ostendorff, Zheng Liu, Simon Clematide, Lester James Miranda, Alena Fenogenova, Guangyu Song, Ruqiya Bin Safi, Wen-Ding Li, Alessia Borghini, Federico Cassano, Hongjin Su, Jimmy Lin, Howard Yen, Lasse Hansen, Sara Hooker, Chenghao Xiao, Vaibhav Adlakha, Orion Weller, Siva Reddy, and Niklas Muennighoff. Mmteb: Massive multilingual text embedding benchmark. *arXiv preprint arXiv:2502.13595*, 2025. doi: 10.48550/arXiv.2502.13595. URL <https://arxiv.org/abs/2502.13595>.
- [73] Jan Engler, Sandipan Sikdar, Marlene Lutz, and Markus Strohmaier. Sensepolar: Word sense aware interpretability for pre-trained contextual word embeddings, 2023. URL <https://arxiv.org/abs/2301.04704>.
- [74] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Jacob Steinhardt, and Aleksander Madry. Identifying statistical bias in dataset replication. In *ICML*, 2020.
- [75] Sabri Eyuboglu, Maya Varma, Khaled Kamal Saab, Jean-Benoit Delbrouck, Christopher Lee-Messer, Jared Dunnmon, James Zou, and Christopher Re. Domino: Discovering systematic errors with cross-modal embeddings. In *ICLR*, 2022.
- [76] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation, 2018. URL <https://arxiv.org/abs/1805.04833>.
- [77] F. Friedrich, P. Schramowski, M. Brack, L. Struppek, D. Hintersdorf, S. Luccioni, and K. Kersting. Fair diffusion: Instructing text-to-image generation models on fairness. *arXiv preprint arXiv:2302.10893*, 2023.
- [78] Chaoyou Fu, Peixian Chen, Yunhang Shen, Yulei Qin, Mengdan Zhang, Xu Lin, Jinrui Yang, Xiawu Zheng, Ke Li, Xing Sun, Yunsheng Wu, and Rongrong Ji. Mme: A comprehensive evaluation benchmark for multimodal large language models, 2024. URL <https://arxiv.org/abs/2306.13394>.

- [79] Yossi Gandelsman, Alexei A. Efros, and Jacob Steinhardt. Interpreting clip’s image representation via text-based decomposition, 2023.
- [80] Yossi Gandelsman, Alexei A. Efros, and Jacob Steinhardt. Interpreting the second-order effects of neurons in clip, 2024.
- [81] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling, 2020. URL <https://arxiv.org/abs/2101.00027>.
- [82] Sebastian Gehrmann, Tosin Adewumi, Karmanya Aggarwal, Pawan Sasanka Ammanamanchi, Anuoluwapo Aremu, Antoine Bosselut, Khyathi Raghavi Chandu, Miruna-Adriana Clinciu, Dipanjan Das, Kaustubh Dhole, Wanyu Du, Esin Durmus, Ondřej Dušek, Chris Chinenye Emezue, Varun Gangal, Cristina Garbacea, Tatsunori Hashimoto, Yufang Hou, Yacine Jernite, Harsh Jhamtani, Yangfeng Ji, Shailza Jolly, Mihir Kale, Dhruv Kumar, Faisal Ladhak, Aman Madaan, Mounica Maddela, Khyati Mahajan, Saad Mahamood, Bodhisattwa Prasad Majumder, Pedro Henrique Martins, Angelina McMillan-Major, Simon Mille, Emiel van Miltenburg, Moin Nadeem, Shashi Narayan, Vitaly Nikolaev, Andre Niyongabo Rubungo, Salomey Osei, Ankur Parikh, Laura Perez-Beltrachini, Niranjana Ramesh Rao, Vikas Raunak, Juan Diego Rodriguez, Sashank Santhanam, João Sedoc, Thibault Selam, Samira Shaikh, Anastasia Shimorina, Marco Antonio Sobrevilla Cabeza, Hendrik Strobelt, Nishant Subramani, Wei Xu, Diyi Yang, Akhila Yerukola, and Jiawei Zhou. The GEM benchmark: Natural language generation, its evaluation and metrics. In Antoine Bosselut, Esin Durmus, Varun Prashant Gangal, Sebastian Gehrmann, Yacine Jernite, Laura Perez-Beltrachini, Samira Shaikh, and Wei Xu, editors, *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*, pages 96–120, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.gem-1.10. URL <https://aclanthology.org/2021.gem-1.10>.
- [83] Team Gemini, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- [84] Google Gemini Team. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities, 2025. URL <https://arxiv.org/abs/2507.06261>.
- [85] S. Ghosh and A. Caliskan. Person==light-skinned, western man, and sexualization of women of color: Stereotypes in stable diffusion. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 6971–6985, 2023.
- [86] Gabriel Goh, Nick Cammarata, Chelsea Voss, Shan Carter, Michael Petrov, Ludwig Schubert, Alec Radford, and Chris Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021.
- [87] Google. Vision ai — cloud vision api. <https://cloud.google.com/vision>. Accessed: [date].

- [88] Google Cloud. Perform automatic side-by-side evaluation, 2024. URL <https://cloud.google.com/vertex-ai/docs/generative-ai/models/side-by-side-eval>.
- [89] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [90] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering, 2017. URL <https://arxiv.org/abs/1612.00837>.
- [91] Paul Grimal, Hervé Le Borgne, Olivier Ferret, and Julien Tourille. Tiam - a metric for evaluating alignment in text-to-image generation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2890–2899, January 2024.
- [92] Maarten Grootendorst. Bertopic: Neural topic modeling with a class-based tf-idf procedure. *arXiv preprint arXiv:2203.05794*, 2022.
- [93] Bryan Guan, Tanya Roosta, Peyman Passban, and Mehdi Rezagholizadeh. The order effect: Investigating prompt sensitivity to input order in llms, 2025. URL <https://arxiv.org/abs/2502.04134>.
- [94] Tianrui Guan, Fuxiao Liu, Xiyang Wu, Ruiqi Xian, Zongxia Li, Xiaoyu Liu, Xijun Wang, Lichang Chen, Furong Huang, Yaser Yacoob, Dinesh Manocha, and Tianyi Zhou. Hallusion-bench: An advanced diagnostic suite for entangled language hallucination and visual illusion in large vision-language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14375–14385, June 2024.
- [95] Biyang Guo, Xin Zhang, Ziyuan Wang, Minqi Jiang, Jinran Nie, Yuxuan Ding, Jianwei Yue, and Yupeng Wu. How close is chatgpt to human experts? comparison corpus, evaluation, and detection. *arXiv preprint arxiv:2301.07597*, 2023.
- [96] Danna Gurari, Qing Li, Abigale J. Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P. Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [97] Danna Gurari, Qing Li, Chi Lin, Yinan Zhao, Anhong Guo, Abigale J. Stangl, and Jeffrey P. Bigham. Vizwiz-priv: A dataset for recognizing the presence and purpose of private visual information in images taken by blind people. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [98] K. Hamidieh, H. Zhang, T. Hartvigsen, and M. Ghassemi. Identifying implicit social biases in vision-language models. *arXiv preprint arXiv:2411.00997*, 2023.
- [99] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979. ISSN 00359254, 14679876. URL <http://www.jstor.org/stable/2346830>.

- [100] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [101] Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with apps. *NeurIPS*, 2021.
- [102] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *ICCV*, 2021.
- [103] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.
- [104] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021. URL <https://doi.org/10.48550/arXiv.2103.03874>.
- [105] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.
- [106] Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *NIPS*, pages 1693–1701, 2015.
- [107] Evan Hernandez, Sarah Schwettmann, David Bau, Teona Bagashvili, Antonio Torralba, and Jacob Andreas. Natural language descriptions of deep visual features. In *ICLR*, 2021.
- [108] J. Hessel, A. Holtzman, M. Forbes, R.L. Bras, and Y. Choi. Clipscore: A reference-free evaluation metric for image captioning. *arXiv preprint arXiv:2104.08718*, 2021.
- [109] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, pages 6626–6637, 2017.
- [110] Yuening Hu, Jordan Boyd-Graber, Brianna Satinoff, and Alison Smith. Interactive topic modeling. *Mach. Learn.*, 95(3):423–469, June 2014. ISSN 0885-6125. doi: 10.1007/s10994-013-5413-0. URL <https://doi.org/10.1007/s10994-013-5413-0>.
- [111] Yushi Hu, Benlin Liu, Jungo Kasai, Yizhong Wang, Mari Ostendorf, Ranjay Krishna, and Noah A Smith. Tifa: Accurate and interpretable text-to-image faithfulness evaluation with question answering. *arXiv preprint arXiv:2303.11897*, 2023.

- [112] Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. The platonic representation hypothesis. In *ICML*, 2024.
- [113] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021. URL <https://doi.org/10.5281/zenodo.5143773>. If you use this software, please cite it as below.
- [114] Cohere Inc. Command-r and command-r+ models. <https://huggingface.co/CohereForAI/c4ai-command-r-v01>, 2023. Accessed: 2024-10-02.
- [115] Phillip Isola, Devi Parikh, Antonio Torralba, and Aude Oliva. Understanding the intrinsic memorability of images. In *NeurIPS*, 2011.
- [116] Phillip Isola, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. What makes an image memorable? In *CVPR*, 2011.
- [117] Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. Camels in a changing climate: Enhancing lm adaptation with tulu 2, 2023.
- [118] Lalit Jain, Kevin Jamieson, Robert Mankoff, Robert Nowak, and Scott Sievert. The New Yorker cartoon caption contest dataset, 2020. URL <https://nextml.github.io/caption-contest-data/>.
- [119] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.
- [120] Naman Jain, Manish Shetty, Tianjun Zhang, King Han, Koushik Sen, and Ion Stoica. R2e: Turning any github repository into a programming agent environment. In *ICML*, 2024.
- [121] Saachi Jain, Hannah Lawrence, Ankur Moitra, and Aleksander Madry. Distilling model failures as directions in latent space. In *ICLR*, 2023.
- [122] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, L  lio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Th  ophile Gervet, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. Mixtral of experts, 2024.
- [123] Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. Scaling sentence embeddings with large language models. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3182–3196, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.181. URL <https://aclanthology.org/2024.findings-emnlp.181/>.

- [124] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations*, 2024. URL <https://openreview.net/forum?id=VTF8yNQM66>.
- [125] Minsuk Kahng, Ian Tenney, Mahima Pushkarna, Michael Xieyang Liu, James Wexler, Emily Reif, Krystal Kallarackal, Minsuk Chang, Michael Terry, and Lucas Dixon. Llm comparator: Visual analytics for side-by-side evaluation of large language models. *ArXiv*, abs/2402.10524, 2024. URL <https://api.semanticscholar.org/CorpusID:267740498>.
- [126] Minsuk Kahng, Ian Tenney, Mahima Pushkarna, Michael Xieyang Liu, James Wexler, Emily Reif, Krystal Kallarackal, Minsuk Chang, Michael Terry, and Lucas Dixon. Llm comparator: Visual analytics for side-by-side evaluation of large language models, 2024. URL <https://arxiv.org/abs/2402.10524>.
- [127] Hao Kang, Tevin Wang, and Chenyan Xiong. Interpret and control dense retrieval with sparse latent features, 2025. URL <https://arxiv.org/abs/2411.00786>.
- [128] Subhash Kantamneni, Joshua Engels, Senthooran Rajamanoharan, Max Tegmark, and Neel Nanda. Are sparse autoencoders useful? a case study in sparse probing, 2025. URL <https://arxiv.org/abs/2502.16681>.
- [129] Marzena Karpinska, Nader Akoury, and Mohit Iyyer. The perils of using Mechanical Turk to evaluate open-ended text generation. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 1265–1285, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.97. URL <https://aclanthology.org/2021.emnlp-main.97>.
- [130] Douwe Kiela, Max Bartolo, Yixin Nie, Divyansh Kaushik, Atticus Geiger, Zhengxuan Wu, Bertie Vidgen, Grusha Prasad, Amanpreet Singh, Pratik Ringshia, Zhiyi Ma, Tristan Thrush, Sebastian Riedel, Zeerak Waseem, Pontus Stenetorp, Robin Jia, Mohit Bansal, Christopher Potts, and Adina Williams. Dynabench: Rethinking benchmarking in nlp. *NAACL*, 2021.
- [131] Hoeseong Kim, Jongseok Kim, Hyungseok Lee, Hyunsung Park, and Gunhee Kim. Viewpoint-agnostic change captioning with cycle consistency. In *ICCV*, 2021.
- [132] Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanus Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pages 5637–5664. PMLR, 2021.
- [133] Olga Kolesnikova. Survey of word co-occurrence measures for collocation detection. *Computacion y Sistemas*, 20:327–344, 09 2016. doi: 10.13053/CyS-20-3-2456.
- [134] Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, et al.

- Openassistant conversations—democratizing large language model alignment. *arXiv preprint arXiv:2304.07327*, 2023.
- [135] Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52, 1955. URL <https://api.semanticscholar.org/CorpusID:9426884>.
 - [136] Haewoon Kwak, Jisun An, Elise Jing, and Yong-Yeol Ahn. Frameaxis: characterizing microframe bias and intensity with word embedding. *PeerJ Computer Science*, 7:e644, July 2021. ISSN 2376-5992. doi: 10.7717/peerj-cs.644. URL <http://dx.doi.org/10.7717/peerj-cs.644>.
 - [137] John Lafferty and David Blei. Correlated topic models. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems*, volume 18. MIT Press, 2005. URL https://proceedings.neurips.cc/paper_files/paper/2005/file/9e82757e9a1c12cb710ad680db11f6f1-Paper.pdf.
 - [138] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. Tülu 3: Pushing frontiers in open language model post-training. 2024.
 - [139] Hugo Laurençon, Andrés Marafioti, Victor Sanh, and Léo Tronchon. Building and better understanding vision-language models: insights and future directions., 2024.
 - [140] Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nv-embed: Improved techniques for training llms as generalist embedding models, 2025. URL <https://arxiv.org/abs/2405.17428>.
 - [141] Jinhyuk Lee, Feiyang Chen, Sahil Dua, Daniel Cer, Madhuri Shanbhogue, Iftekhhar Naim, Gustavo Hernández Ábrego, Zhe Li, Kaifeng Chen, Henrique Schechter Vera, Xiaoqi Ren, Shanfeng Zhang, Daniel Salz, Michael Boratko, Jay Han, Blair Chen, Shuo Huang, Vikram Rao, Paul Suganthan, Feng Han, Andreas Dourmoglou, Nithi Gupta, Fedor Moiseev, Cathy Yip, Aashi Jain, Simon Baumgartner, Shahrokh Shahi, Frank Palma Gomez, Sandeep Mariserla, Min Choi, Parashar Shah, Sonam Goenka, Ke Chen, Ye Xia, Koert Chen, Sai Meher Karthik Duddu, Yichang Chen, Trevor Walker, Wenlei Zhou, Rakesh Ghiya, Zach Gleicher, Karan Gill, Zhe Dong, Mojtaba Seyedhosseini, Yunhsuan Sung, Raphael Hoffmann, and Tom Duerig. Gemini embedding: Generalizable embeddings from gemini, 2025. URL <https://arxiv.org/abs/2503.07891>.
 - [142] Minchul Lee. bab2min/tomotopy: 0.12.3, July 2022. URL <https://doi.org/10.5281/zenodo.6868418>.
 - [143] Seongyun Lee, Seungone Kim, Sue Hyun Park, Geewook Kim, and Minjoon Seo. Prometheus-vision: Vision-language model as a judge for fine-grained evaluation, 2024. URL <https://arxiv.org/abs/2401.06591>.

- [144] Tony Lee, Michihiro Yasunaga, Chenlin Meng, Yifan Mai, Joon Sung Park, Agrim Gupta, Yunzhi Zhang, Deepak Narayanan, Hannah Benita Teufel, Marco Bellagente, Minguk Kang, Taesung Park, Jure Leskovec, Jun-Yan Zhu, Li Fei-Fei, Jiajun Wu, Stefano Ermon, and Percy Liang. Holistic evaluation of text-to-image models. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023.
- [145] Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Fanyi Pu, Jingkang Yang, Chunyuan Li, and Ziwei Liu. Mimic-it: Multi-modal in-context instruction tuning. *arXiv preprint arXiv:2306.05425*, 2023.
- [146] Bo Li, Yuanhan Zhang, Liangyu Chen, Jinghao Wang, Jingkang Yang, and Ziwei Liu. Otter: A multi-modal model with in-context instruction tuning. *arXiv preprint arXiv:2305.03726*, 2023.
- [147] Daiqing Li, Aleks Kamko, Ehsan Akhgari, Ali Sabet, Linmiao Xu, and Suhail Doshi. Playground v2.5: Three insights towards enhancing aesthetic quality in text-to-image generation, 2024.
- [148] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.
- [149] Margaret Li, Jason Weston, and Stephen Roller. Acute-eval: Improved dialogue evaluation with optimized questions and multi-turn comparisons, 2019. URL <https://arxiv.org/abs/1909.03087>.
- [150] Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline, 2024. URL <https://arxiv.org/abs/2406.11939>.
- [151] Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Alpaca-eval: An automatic evaluator of instruction-following models. https://github.com/tatsu-lab/alpaca_eval, 2023.
- [152] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *arXiv preprint arXiv:2211.09110*, 2022.
- [153] Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, Benjamin Newman, Binhang Yuan, Bobby Yan, Ce Zhang, Christian Cosgrove, Christopher D. Manning, Christopher Ré, Diana Acosta-Navas, Drew A. Hudson, Eric Zelikman, Esin Durmus, Faisal Ladhak, Frieda Rong, Hongyu Ren, Huaxiu Yao, Jue Wang, Keshav Santhanam, Laurel Orr, Lucia Zheng, Mert Yuksekgonul, Mirac Suzgun, Nathan Kim, Neel Guha, Niladri Chatterji, Omar Khattab, Peter Henderson, Qian Huang, Ryan Chi, Sang Michael Xie, Shibani Santurkar, Surya Ganguli, Tatsunori Hashimoto, Thomas Icard, Tianyi Zhang, Vishrav Chaudhary, William Wang, Xuechen Li, Yifan Mai, Yuhui Zhang, and Yuta Koreeda. Holistic evaluation of language models. *Transactions on Machine Learning Research*, 08 2023.

- [154] Weixin Liang and James Zou. Metashift: A dataset of datasets for evaluating contextual distribution shifts and training conflicts. In *ICLR*, 2022.
- [155] Rensis Likert. A technique for the measurement of attitudes. *Archives of psychology*, 1932.
- [156] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL <https://aclanthology.org/W04-1013>.
- [157] Shanchuan Lin, Anran Wang, and Xiao Yang. Sdxl-lightning: Progressive adversarial diffusion distillation, 2024.
- [158] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, C Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context. *European conference on computer vision*, pages 740–755, 2014.
- [159] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.
- [160] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.
- [161] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning, 2023. URL <https://arxiv.org/abs/2304.08485>.
- [162] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024. URL <https://llava-vl.github.io/blog/2024-01-30-llava-next/>.
- [163] Mingxuan Liu, Zhun Zhong, Jun Li, Gianni Franchi, Subhankar Roy, and Elisa Ricci. Organizing unstructured image collections using natural language. *arXiv preprint arXiv:2410.05217*, 2024. URL <https://arxiv.org/abs/2410.05217>.
- [164] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *arXiv preprint arXiv:2307.03172*, 2023.
- [165] Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, Minlie Huang, Yuxiao Dong, and Jie Tang. Agentbench: Evaluating llms as agents, 2023.
- [166] Adian Liusie, Potsawee Manakul, and Mark J. F. Gales. Llm comparative assessment: Zero-shot nlg evaluation through pairwise comparisons using large language models. *EACL*, 2024.
- [167] Yujie* Lu, Dongfu* Jiang, Hui* Chen, Xingyu Fu, Yingzi Ma, Jing Gu, Michael Saxon, Chaowei Xiao, Wenhui Chen, Yejin Choi, Bill Yuchen Lin, Miguel Eckstein, and William Wang. Wildvision data and model, 2024. URL <https://huggingface.co/WildVision>.

- [168] Yujie Lu, Dongfu Jiang, Wenhui Chen, William Yang Wang, Yejin Choi, and Bill Yuchen Lin. Wildvision: Evaluating vision-language models in the wild with human preferences. 2024.
- [169] Alexandra Sasha Luccioni, Cynthia Akiki, Margaret Mitchell, and Yacine Jernite. Stable bias: Analyzing societal representations in diffusion models. In *Proceedings of the 37th Conference on Neural Information Processing Systems (NeurIPS)*, 2023.
- [170] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *NeurIPS*, 2017.
- [171] Xing Han Lù. Bm25s: Orders of magnitude faster lexical search via eager sparse scoring, 2024. URL <https://arxiv.org/abs/2407.03618>.
- [172] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL <http://www.aclweb.org/anthology/P11-1015>.
- [173] Claudia Malzer and Marcus Baum. A hybrid approach to hierarchical density-based cluster selection. In *2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, page 223–228. IEEE, September 2020. doi: 10.1109/mfi49285.2020.9235263. URL <http://dx.doi.org/10.1109/MFI49285.2020.9235263>.
- [174] Lev Manovich. *How to Compare One Million Images?*, pages 249–278. Palgrave Macmillan UK, London, 2012. ISBN 978-0-230-37193-4. doi: 10.1057/9780230371934_14. URL https://doi.org/10.1057/9780230371934_14.
- [175] Ahmed Masry, Do Long, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. ChartQA: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2263–2279, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.177. URL <https://aclanthology.org/2022.findings-acl.177>.
- [176] Binny Mathew, Sandipan Sikdar, Florian Lemmerich, and Markus Strohmaier. The polar framework: Polar opposites enable interpretability of pre-trained word embeddings, 2020. URL <https://arxiv.org/abs/2001.09876>.
- [177] Minesh Mathew, Dimosthenis Karatzas, R Manmatha, and CV Jawahar. Docvqa: A dataset for vqa on document images. corr abs/2007.00398 (2020). *arXiv preprint arXiv:2007.00398*, 2020.
- [178] Thomas McGrath, Daniel Balsam, Myra Deng, and Eric Ho. Understanding and steering llama 3 with sparse autoencoders, 2024. URL <https://www.goodfire.ai/research/understanding-and-steering-llama-3>.
- [179] Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205, 2017. doi: 10.21105/joss.00205. URL <https://doi.org/10.21105/joss.00205>.

- [180] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction, 2020. URL <https://arxiv.org/abs/1802.03426>.
- [181] Shikib Mehri and Maxine Eskenazi. Unsupervised evaluation of interactive dialog with DialoGPT. In Olivier Pietquin, Smaranda Muresan, Vivian Chen, Casey Kennington, David Vandyke, Nina Dethlefs, Koji Inoue, Erik Ekstedt, and Stefan Ultes, editors, *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 225–235, 1st virtual meeting, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.sigdial-1.28. URL <https://aclanthology.org/2020.sigdial-1.28>.
- [182] Shikib Mehri and Maxine Eskenazi. USR: An unsupervised and reference free evaluation metric for dialog generation. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 681–707, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.64. URL <https://aclanthology.org/2020.acl-main.64>.
- [183] Kevin Meng, Vincent Huang, Jacob Steinhardt, and Sarah Schwettmann. Introducing docent. <https://transluce.org/introducing-docent>, March 2025.
- [184] Meta AI. Llama 3.3 model card. https://github.com/meta-llama/llama-models/blob/main/models/llama3_3/MODEL_CARD.md, 2024. Accessed: 2025-08-04.
- [185] Microsoft. Photodna service. Online, 2024. URL <https://www.microsoft.com/en-us/photodna>. Accessed: 2024-11-18.
- [186] John P Miller, Rohan Taori, Aditi Raghunathan, Shiori Sagawa, Pang Wei Koh, Vaishaal Shankar, Percy Liang, Yair Carmon, and Ludwig Schmidt. Accuracy on the line: on the strong correlation between out-of-distribution and in-distribution generalization. In *ICML*, 2021.
- [187] Marco Molinari, Victor Shao, Luca Imeneo, Mateusz Mikolajczak, Vladimir Tregubiak, Abhimanyu Pandey, and Sebastian Kuznetsov Ryder Torres Pereira. Interpretable company similarity with sparse autoencoders, 2025. URL <https://arxiv.org/abs/2412.02605>.
- [188] Rajiv Movva, Kenny Peng, Nikhil Garg, Jon Kleinberg, and Emma Pierson. Sparse autoencoders for hypothesis generation, 2025. URL <https://arxiv.org/abs/2502.04382>.
- [189] Niklas Muennighoff. Sgpt: Gpt sentence embeddings for semantic search, 2022. URL <https://arxiv.org/abs/2202.08904>.
- [190] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022. doi: 10.48550/ARXIV.2210.07316. URL <https://arxiv.org/abs/2210.07316>.
- [191] Joel Nothman, Nicky Ringland, Will Radford, Tara Murphy, and James R. Curran. Learning multilingual named entity recognition from wikipedia. *Artificial Intelligence*, 194: 151–175, 2013. ISSN 0004-3702. doi: <https://doi.org/10.1016/j.artint.2012.03.006>. URL <https://www.sciencedirect.com/science/article/pii/S0004370212000276>. Artificial Intelligence, Wikipedia and Semi-Structured Resources.

- [192] Charles O’Neill, Christine Ye, Kartheik Iyer, and John F. Wu. Disentangling dense embeddings with sparse autoencoders, 2024. URL <https://arxiv.org/abs/2408.00657>.
- [193] OpenAI. Gpt-4 technical report, 2023.
- [194] OpenAI. New models and developer products announced at devday. <https://openai.com/blog/new-models-and-developer-products-announced-at-devday>, 2023. (Accessed on 06/05/2024).
- [195] OpenAI. Openai text moderation tool. Online, 2024. URL <https://platform.openai.com/docs/guides/moderation>. Accessed: 2024-11-18.
- [196] OpenAI. New embedding models and api updates. <https://openai.com/index/new-embedding-models-and-api-updates/>, 2024. (Accessed on 06/05/2024).
- [197] OpenAI. Hello gpt-4o. <https://openai.com/index/hello-gpt-4o/>, 2024. (Accessed on 06/05/2024).
- [198] OpenAI. Openai embeddings. <https://platform.openai.com/docs/guides/embeddings>, 2024. Accessed: July 2025.
- [199] Piotr Padlewski, Max Bain, Matthew Henderson, Zhongkai Zhu, Nishant Relan, Hai Pham, Donovan Ong, Kaloyan Aleksiev, Aitor Ormazabal, Samuel Phua, Ethan Yeo, Eugenie Lamprecht, Qi Liu, Yuqi Wang, Eric Chen, Deyu Fu, Lei Li, Che Zheng, Cyprien de Masson d’Autume, Dani Yogatama, Mikel Artetxe, and Yi Tay. Vibe-eval: A hard evaluation suite for measuring progress of multimodal language models, 2024.
- [200] Bo Pang, Erik Nijkamp, Wenjuan Han, Linqi Zhou, Yixian Liu, and Kewei Tu. Towards holistic and automatic evaluation of open-domain dialogue generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3619–3629, Online, 2020. Association for Computational Linguistics.
- [201] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, 2002. Association for Computational Linguistics.
- [202] ChaeHun Park, Minseok Choi, Dohyun Lee, and Jaegul Choo. Paireval: Open-domain dialogue evaluation with pairwise comparison. *arXiv preprint arXiv:2404.01015*, 2024.
- [203] Dong Huk Park, Trevor Darrell, and Anna Rohrbach. Robust change captioning. In *ICCV*, 2019.
- [204] Dong Huk Park, Samaneh Azadi, Xihui Liu, Trevor Darrell, and Anna Rohrbach. Benchmark for compositional text-to-image synthesis. In *Proceedings of the Neural Information Processing Systems (NeurIPS)*. University of California, Berkeley, 2024. URL <https://github.com/Seth-Park/comp-t2i-dataset>.

- [205] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Madry. Trak: Attributing model behavior at scale. In *ICML*, 2023.
- [206] Liana Patel, Siddharth Jha, Melissa Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. Semantic operators: A declarative model for rich, ai-based data processing, 2025. URL <https://arxiv.org/abs/2407.11418>.
- [207] Gonalo Paulo, Alex Mallen, Caden Juang, and Nora Belrose. Automatically interpreting millions of features in large language models, 2024. URL <https://arxiv.org/abs/2410.13928>.
- [208] Nick Craswell Li Deng Jianfeng Gao Xiaodong Liu Rangan Majumder Andrew McNamara Bhaskar Mitra Tri Nguyen Mir Rosenberg Xia Song Alina Stoica Saurabh Tiwary Tong Wang Payal Bajaj, Daniel Campos. Ms marco: A human generated machine reading comprehension dataset. In *InCoCo@NIPS*, 2016.
- [209] Karl Pearson. Note on regression and inheritance in the case of two parents. *Proceedings of the Royal Society of London*, 58:240–242, 1895. ISSN 03701662. URL <http://www.jstor.org/stable/115794>.
- [210] Joaquin Quinonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. Mit Press, 2008.
- [211] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR, 18–24 Jul 2021.
- [212] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision, 2021. URL <https://arxiv.org/abs/2103.00020>.
- [213] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.
- [214] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *EMNLP*, 2016.
- [215] Shauli Ravfogel, Valentina Pyatkin, Amir DN Cohen, Avshalom Manevich, and Yoav Goldberg. Description-based text similarity, 2024. URL <https://arxiv.org/abs/2305.12517>.
- [216] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *ICML*, 2019.

- [217] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016.
- [218] Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.
- [219] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019. URL <https://arxiv.org/abs/1908.10084>.
- [220] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023.
- [221] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?" explaining the predictions of any classifier. In *KDD*, 2016.
- [222] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3:333–389, 01 2009. doi: 10.1561/15000000019.
- [223] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022.
- [224] Sara Rosenthal, Noura Farra, and Preslav Nakov. Semeval-2017 task 4: Sentiment analysis in twitter. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)*, pages 502–518, 2017.
- [225] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karagol Ayan, S Sara Mahdavi, Rapha Gontijo Lopes, et al. Photorealistic text-to-image diffusion models with deep language understanding. *arXiv preprint arXiv:2205.11487*, 2022.
- [226] Oscar Sainz, Jon Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10776–10787, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.722. URL <https://aclanthology.org/2023.findings-emnlp.722>.
- [227] Keita Saito, Akifumi Wachi, Koki Wataoka, and Youhei Akimoto. Verbosity bias in preference labeling by large language models, 2023. URL <https://arxiv.org/abs/2310.10076>.
- [228] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 29, pages 2234–2242, 2016.

- [229] Gerard Salton and Chris Buckley. Term weighting approaches in automatic text retrieval. Technical report, USA, 1987.
- [230] Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Commun. ACM*, 18:613–620, 1975. URL <https://api.semanticscholar.org/CorpusID:6473756>.
- [231] Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3687–3697, Brussels, Belgium, October–November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1404. URL <https://www.aclweb.org/anthology/D18-1404>.
- [232] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL <https://openreview.net/forum?id=M3Y74vmsMcY>.
- [233] John Schulman, Barret Zoph, Christina Kim, Jacob Hilton, Jacob Menick, Jiayi Weng, Juan Felipe Ceron Uribe, Liam Fedus, Luke Metz, Michael Pokorny, et al. Chatgpt: Optimizing language models for dialogue. *OpenAI blog*, 2022.
- [234] Thibault Sellam, Dipanjan Das, and Ankur P Parikh. Bleurt: Learning robust metrics for text generation. In *Proceedings of ACL*, 2020. URL <https://arxiv.org/abs/2004.04696>.
- [235] Nimrod Shabtay, Felipe Maia Polo, Sivan Doveh, Wei Lin, M. Jehanzeb Mirza, Leshem Chosen, Mikhail Yurochkin, Yuekai Sun, Assaf Arbelle, Leonid Karlinsky, and Raja Giryes. Livexiv – a multi-modal live benchmark based on arxiv papers content, 2024. URL <https://arxiv.org/abs/2410.10783>.
- [236] Harshay Shah, Sung Min Park, Andrew Ilyas, and Aleksander Madry. Modeldiff: A framework for comparing learning algorithms. In *ICML*, 2023.
- [237] Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, and Eugene Wu. Docetl: Agentic query rewriting and evaluation for complex document processing, 2025. URL <https://arxiv.org/abs/2410.12189>.
- [238] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. *arXiv preprint arXiv:2302.00093*, 2023.
- [239] Amanpreet Singh, Vivek Natarjan, Meet Shah, Yu Jiang, Xinlei Chen, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8317–8326, 2019.

- [240] Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhunoye, George Zerveas, Vijay Korthikanti, Elton Zheng, Rewon Child, Reza Yazdani Aminabadi, Julie Bernauer, Xia Song, Mohammad Shoeybi, Yuxiong He, Michael Houston, Saurabh Tiwary, and Bryan Catanzaro. Using deep-speed and megatron to train megatron-turing NLG 530b, A large-scale generative language model. *CoRR*, abs/2201.11990, 2022. URL <https://arxiv.org/abs/2201.11990>.
- [241] Karen Spärck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28(1):11–21, 1972.
- [242] Charles Spearman. The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101, January 1904. doi: 10.2307/1412159. URL <https://doi.org/10.2307/1412159>.
- [243] AaroHi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on Machine Learning Research*, 2023.
- [244] Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. One embedder, any task: Instruction-finetuned text embeddings. 2022. URL <https://arxiv.org/abs/2212.09741>.
- [245] Mingjie Sun, Yida Yin, Zhiqiu Xu, J. Zico Kolter, and Zhuang Liu. Idiosyncrasies in large language models, 2025. URL <https://arxiv.org/abs/2502.12150>.
- [246] Wei Ren Tan, Chee Seng Chan, Hernan Aguirre, and Kiyoshi Tanaka. Improved artgan for conditional synthesis of natural image and artwork. *IEEE Transactions on Image Processing*, 28(1):394–409, 2019. doi: 10.1109/TIP.2018.2866698. URL <https://doi.org/10.1109/TIP.2018.2866698>.
- [247] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.
- [248] Gemini Team, Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, Ioannis Antonoglou, Rohan Anil, Sebastian Borgeaud, Andrew Dai, Katie Millican, Ethan Dyer, Mia Glaese, Thibault Sottiaux, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, James Molloy, Jilin Chen, Michael Isard, Paul Barham, Tom Hennigan, Ross McIlroy, Melvin Johnson, Johan Schalkwyk, Eli Collins, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, Clemens Meyer, Gregory Thornton, Zhen Yang, Henryk Michalewski, Zaheer Abbas, Nathan Schucher, Ankesh Anand, Richard Ives, James Keeling, Karel Lenc, Salem Haykal, Siamak Shakeri, Pranav Shyam, Aakanksha Chowdhery, Roman Ring, Stephen Spencer, Eren Sezener, Luke Vilnis, Oscar Chang, Nobuyuki Morioka, George Tucker, Ce Zheng, Oliver Woodman, Nithya Attaluri, Tomas Kocisky, Evgenii Eltyshhev, Xi Chen, Timothy Chung, Vittorio Selo, Siddhartha Brahma, Petko Georgiev,

Ambrose Slone, Zhenkai Zhu, James Lottes, Siyuan Qiao, Ben Caine, Sebastian Riedel, Alex Tomala, Martin Chadwick, Juliette Love, Peter Choy, Sid Mittal, Neil Houlsby, Yunhao Tang, Matthew Lamm, Libin Bai, Qiao Zhang, Luheng He, Yong Cheng, Peter Humphreys, Yujia Li, Sergey Brin, Albin Cassirer, Yingjie Miao, Lukas Zilka, Taylor Tobin, Kelvin Xu, Lev Proleev, Daniel Sohn, Alberto Magni, Lisa Anne Hendricks, Isabel Gao, Santiago Ontanon, Oskar Bunyan, Nathan Byrd, Abhanshu Sharma, Biao Zhang, Mario Pinto, Rishika Sinha, Harsh Mehta, Dawei Jia, Sergi Caelles, Albert Webson, Alex Morris, Becca Roelofs, Yifan Ding, Robin Strudel, Xuehan Xiong, Marvin Ritter, Mostafa Dehghani, Rahma Chaabouni, Abhijit Karmarkar, Guangda Lai, Fabian Mentzer, Bibo Xu, YaGuang Li, Yujing Zhang, Tom Le Paine, Alex Goldin, Behnam Neyshabur, Kate Baumli, Anselm Levskaya, Michael Laskin, Wenhao Jia, Jack W. Rae, Kefan Xiao, Antoine He, Skye Girdano, Lakshman Yagati, Jean-Baptiste Lespiau, Paul Natsev, Sanjay Ganapathy, Fangyu Liu, Danilo Martins, Nanxin Chen, Yunhan Xu, Megan Barnes, Rhys May, Arpi Vezer, Junhyuk Oh, Ken Franko, Sophie Bridgers, Ruizhe Zhao, Boxi Wu, Basil Mustafa, Sean Sechrist, Emilio Parisotto, Thanumalayan Sankaranarayanan Pillai, Chris Larkin, Chenjie Gu, Christina Sorokin, Maxim Krikun, Alexey Guseynov, Jessica Landon, Romina Datta, Alexander Pritzel, Phoebe Thacker, Fan Yang, Kevin Hui, Anja Hauth, Chih-Kuan Yeh, David Barker, Justin Mao-Jones, Sophia Austin, Hannah Sheahan, Parker Schuh, James Svensson, Rohan Jain, Vinay Ramasesh, Anton Briukhov, Da-Woon Chung, Tamara von Glehn, Christina Butterfield, Priya Jhakra, Matthew Wiethoff, Justin Frye, Jordan Grinstead, Beer Changpinyo, Charline Le Lan, Anna Bortsova, Yonghui Wu, Paul Voigtlaender, Tara Sainath, Shane Gu, Charlotte Smith, Will Hawkins, Kris Cao, James Besley, Srivatsan Srinivasan, Mark Omernick, Colin Gaffney, Gabriela Surita, Ryan Burnell, Bogdan Damoc, Junwhan Ahn, Andrew Brock, Mantas Pajarskas, Anastasia Petrushkina, Seb Noury, Lorenzo Blanco, Kevin Swersky, Arun Ahuja, Thi Avrahami, Vedant Misra, Raoul de Liedekerke, Mariko Iinuma, Alex Polozov, Sarah York, George van den Driessche, Paul Michel, Justin Chiu, Rory Blevins, Zach Gleicher, Adrià Recasens, Alban Rustemi, Elena Gribovskaya, Aurko Roy, Wiktor Gworek, Sébastien M. R. Arnold, Lisa Lee, James Lee-Thorp, Marcello Maggioni, Enrique Piqueras, Kartikeya Badola, Sharad Vikram, Lucas Gonzalez, Anirudh Baddepudi, Evan Senter, Jacob Devlin, James Qin, Michael Azzam, Maja Trebacz, Martin Polacek, Kashyap Krishnakumar, Shuo Yin Chang, Matthew Tung, Ivo Penchev, Rishabh Joshi, Kate Olszewska, Carrie Muir, Mateo Wirth, Ale Jakse Hartman, Josh Newlan, Sheleem Kashem, Vijay Bolina, Elahe Dabir, Joost van Amersfoort, Zafarali Ahmed, James Cobon-Kerr, Aishwarya Kamath, Arnar Mar Hrafnkelsson, Le Hou, Ian Mackinnon, Alexandre Frechette, Eric Noland, Xiance Si, Emanuel Taropa, Dong Li, Phil Crone, Anmol Gulati, Sébastien Cevey, Jonas Adler, Ada Ma, David Silver, Simon Tokumine, Richard Powell, Stephan Lee, Kiran Vodrahalli, Samer Hassan, Diana Mincu, Antoine Yang, Nir Levine, Jenny Brennan, Mingqiu Wang, Sarah Hodgkinson, Jeffrey Zhao, Josh Lipschultz, Aedan Pope, Michael B. Chang, Cheng Li, Laurent El Shafey, Michela Paganini, Sholto Douglas, Bernd Bohnet, Fabio Pardo, Seth Odoom, Mihaela Rosca, Cicero Nogueira dos Santos, Kedar Soparkar, Arthur Guez, Tom Hudson, Steven Hansen, Chulayuth Asawaroengchai, Ravi Addanki, Tianhe Yu, Wojciech Stokowiec, Mina Khan, Justin Gilmer, Jaehoon Lee, Carrie Grimes Bostock, Keran Rong, Jonathan Caton, Pedram Pejman, Filip Pavetic, Geoff Brown, Vivek Sharma, Mario Lučić, Rajkumar Samuel, Josip Djolonga, Amol Mandhane,

Lars Lowe Sjöstrand, Elena Buchatskaya, Elspeth White, Natalie Clay, Jiepu Jiang, Hyeontaek Lim, Ross Hemsley, Zeyncep Cankara, Jane Labanowski, Nicola De Cao, David Steiner, Sayed Hadi Hashemi, Jacob Austin, Anita Gergely, Tim Blyth, Joe Stanton, Kaushik Shivakumar, Aditya Siddhant, Anders Andreassen, Carlos Araya, Nikhil Sethi, Rakesh Shivanna, Steven Hand, Ankur Bapna, Ali Khodaei, Antoine Miech, Garrett Tanzer, Andy Swing, Shantanu Thakoor, Lora Aroyo, Zhufeng Pan, Zachary Nado, Jakub Sygnowski, Stephanie Winkler, Dian Yu, Mohammad Saleh, Loren Maggiore, Yamini Bansal, Xavier Garcia, Mehran Kazemi, Piyush Patil, Ishita Dasgupta, Iain Barr, Minh Giang, Thais Kagohara, Ivo Danihelka, Amit Marathe, Vladimir Feinberg, Mohamed Elhawaty, Nimesh Ghelani, Dan Horgan, Helen Miller, Lexi Walker, Richard Tanburn, Mukarram Tariq, Disha Shrivastava, Fei Xia, Qingze Wang, Chung-Cheng Chiu, Zoe Ashwood, Khuslen Baatarsukh, Sina Samangooei, Raphaël Lopez Kaufman, Fred Alcober, Axel Stjerngren, Paul Komarek, Katerina Tsihlias, Anudhyan Boral, Ramona Comanescu, Jeremy Chen, Ruibo Liu, Chris Welty, Dawn Bloxwich, Charlie Chen, Yanhua Sun, Fangxiaoyu Feng, Matthew Mauger, Xerxes Dotiwalla, Vincent Hellendoorn, Michael Sharman, Ivy Zheng, Krishna Haridasan, Gabe Barth-Maron, Craig Swanson, Dominika Rogozińska, Alek Andreev, Paul Kishan Rubenstein, Ruoxin Sang, Dan Hurt, Gamaleldin Elsayed, Renshen Wang, Dave Lacey, Anastasija Ilić, Yao Zhao, Adam Iwanicki, Alejandro Lince, Alexander Chen, Christina Lyu, Carl Lebsack, Jordan Griffith, Meenu Gaba, Paramjit Sandhu, Phil Chen, Anna Koop, Ravi Rajwar, Soheil Hassas Yeganeh, Solomon Chang, Rui Zhu, Soroush Radpour, Elnaz Davoodi, Ving Ian Lei, Yang Xu, Daniel Toyama, Constant Segal, Martin Wicke, Hanzhao Lin, Anna Bulanova, Adrià Puigdomènech Badia, Nemanja Rakićević, Pablo Sprechmann, Angelos Filos, Shaobo Hou, Víctor Campos, Nora Kassner, Devendra Sachan, Meire Fortunato, Chimezie Iwuanyanwu, Vitaly Nikolaev, Balaji Lakshminarayanan, Sadegh Jazayeri, Mani Varadarajan, Chetan Tekur, Doug Fritz, Misha Khalman, David Reitter, Kingshuk Dasgupta, Shourya Sarcar, Tina Ornduff, Javier Snider, Fantine Huot, Johnson Jia, Rupert Kemp, Nejc Trdin, Anitha Vijayakumar, Lucy Kim, Christof Angermueller, Li Lao, Tianqi Liu, Haibin Zhang, David Engel, Somer Greene, Anaïs White, Jessica Austin, Lilly Taylor, Shereen Ashraf, Dangyi Liu, Maria Georgaki, Irene Cai, Yana Kulizhskaya, Sonam Goenka, Brennan Saeta, Ying Xu, Christian Frank, Dario de Cesare, Brona Robenek, Harry Richardson, Mahmoud Alnahlawi, Christopher Yew, Priya Ponnampalli, Marco Tagliasacchi, Alex Korchemniy, Yelin Kim, Dinghua Li, Bill Rosgen, Kyle Levin, Jeremy Wiesner, Praseem Banzal, Praveen Srinivasan, Hongkun Yu, Çağlar Ünlü, David Reid, Zora Tung, Daniel Finchelstein, Ravin Kumar, Andre Elisseeff, Jin Huang, Ming Zhang, Ricardo Aguilar, Mai Giménez, Jiawei Xia, Olivier Dousse, Willi Gierke, Damion Yates, Komal Jalan, Lu Li, Eri Latorre-Chimoto, Duc Dung Nguyen, Ken Durden, Praveen Kallakuri, Yaxin Liu, Matthew Johnson, Tomy Tsai, Alice Talbert, Jasmine Liu, Alexander Neitz, Chen Elkind, Marco Selvi, Mimi Jasarevic, Livio Baldini Soares, Albert Cui, Pidong Wang, Alek Wenjiao Wang, Xinyu Ye, Krystal Kallarakal, Lucia Loher, Hoi Lam, Josef Broder, Dan Holtmann-Rice, Nina Martin, Bramandia Ramadhana, Mrinal Shukla, Sujoy Basu, Abhi Mohan, Nick Fernando, Noah Fiedel, Kim Paterson, Hui Li, Ankush Garg, Jane Park, DongHyun Choi, Diane Wu, Sankalp Singh, Zhishuai Zhang, Amir Globerson, Lily Yu, John Carpenter, Félix de Chaumont Quitry, Carey Radebaugh, Chu-Cheng Lin, Alex Tudor, Prakash Shroff, Drew Garmon, Dayou Du, Neera Vats, Han Lu, Shariq Iqbal, Alex Yakubovich, Nilesch Tripuraneni, James Manyika, Haroon Qureshi, Nan Hua, Christel Ngani,

- Maria Abi Raad, Hannah Forbes, Jeff Stanway, Mukund Sundararajan, Victor Ungureanu, Colton Bishop, Yunjie Li, Balaji Venkatraman, Bo Li, Chloe Thornton, Salvatore Scellato, Nishesh Gupta, Yicheng Wang, Ian Tenney, Xihui Wu, Ashish Shenoy, Gabriel Carvajal, Diana Gage Wright, Ben Bariach, Zhuyun Xiao, Peter Hawkins, Sid Dalmia, Clement Farabet, Pedro Valenzuela, Quan Yuan, Ananth Agarwal, Mia Chen, Wooyeol Kim, Brice Hulse, Nandita Dukkupati, Adam Paszke, Andrew Bolt, Kiam Choo, Jennifer Beattie, Jennifer Prendki, Harsha Vashisht, Rebeca Santamaria-Fernandez, Luis C. Cobo, Jarek Wilkiewicz, David Madras, Ali Elqursh, Grant Uy, Kevin Ramirez, Matt Harvey, Tyler Liechty, Heiga Zen, Jeff Seibert, Clara Huiyi Hu, Andrey Khorlin, Maigo Le, Asaf Aharoni, Megan Li, Lily Wang, Sandeep Kumar, Norman Casagrande, Jay Hoover, Dalia El Badawy, David Soergel, Denis Vnukov, Matt Miecnikowski, Jiri Simsa, Praveen Kumar, Thibault Sellam, Daniel Vlasic, Samira Daruki, Nir Shabat, John Zhang, Guolong Su, Jiageng Zhang, Jeremiah Liu, Yi Sun, Evan Palmer, Alireza Ghaffarkhah, Xi Xiong, Victor Cotruta, Michael Fink, Lucas Dixon, Ashwin Sreevatsa, Adrian Goedeckemeyer, Alek Dimitriev, Mohsen Jafari, Remi Crocker, Nicholas FitzGerald, Aviral Kumar, Sanjay Ghemawat, Ivan Philips, Frederick Liu, Yannie Liang, Rachel Sterneck, Alena Repina, Marcus Wu, Laura Knight, Marin Georgiev, Hyo Lee, Harry Askham, Abhishek Chakladar, Annie Louis, Carl Crous, Hardie Cate, Dessie Petrova, Michael Quinn, Denese Owusu-Afriyie, Achintya Singhal, Nan Wei, Solomon Kim, Damien Vincent, Milad Nasr, Christopher A. Choquette-Choo, Reiko Tojo, Shawn Lu, Diego de Las Casas, Yuchung Cheng, Tolga Bolukbasi, Katherine Lee, Saaber Fatehi, Rajagopal Ananthanarayanan, Miteyan Patel, Charbel Kaed, Jing Li, Shreyas Rammohan Belle, Zhe Chen, Jaclyn Konzelmann, Siim Pöder, Roopal Garg, Vinod Koverkathu, Adam Brown, Chris Dyer, Rosanne Liu, Azade Nova, Jun Xu, Alanna Walton, Alicia Parrish, Mark Epstein, Sara McCarthy, Slav Petrov, Demis Hassabis, Koray Kavukcuoglu, Jeffrey Dean, and Oriol Vinyals. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, 2024.
- [249] Mistral AI team. Au large. <https://mistral.ai/news/mistral-large/>, 2024. (Accessed on 06/05/2024).
- [250] Qwen Team. Introducing qwen1.5, February 2024. URL <https://qwenlm.github.io/blog/qwen1.5/>.
- [251] Reka Team, Aitor Ormazabal, Che Zheng, Cyprien de Masson d’Autume, Dani Yogatama, Deyu Fu, Donovan Ong, Eric Chen, Eugenie Lamprecht, Hai Pham, Isaac Ong, Kaloyan Aleksiev, Lei Li, Matthew Henderson, Max Bain, Mikel Artetxe, Nishant Relan, Piotr Padlewski, Qi Liu, Ren Chen, Samuel Phua, Yazheng Yang, Yi Tay, Yuqi Wang, Zhongkai Zhu, and Zhihui Xie. Reka core, flash, and edge: A series of powerful multimodal language models, 2024. URL <https://arxiv.org/abs/2404.12387>.
- [252] The Mosaic Research Team. Introducing dbrx: A new state-of-the-art open llm. <https://www.databricks.com/blog/introducing-dbrx-new-state-art-open-llm/>, 2024. (Accessed on 06/05/2024).
- [253] Adly Templeton, Tom Conerly, Jonathan Marcus, Jack Lindsey, Trenton Bricken, Brian Chen, Adam Pearce, Craig Citro, Emmanuel Ameisen, Andy Jones, Hoagy Cunningham,

- Nicholas L Turner, Callum McDougall, Monte MacDiarmid, C. Daniel Freeman, Theodore R. Sumers, Edward Rees, Joshua Batson, Adam Jermyn, Shan Carter, Chris Olah, and Tom Henighan. Scaling monosemanticity: Extracting interpretable features from claude 3 sonnet. *Transformer Circuits Thread*, 2024. URL <https://transformer-circuits.pub/2024/scaling-monosemanticity/index.html>.
- [254] Claire Tian, Katherine Tian, and Nathan Hu. Measuring sparse autoencoder feature sensitivity. URL <https://openreview.net/forum?id=0bkTiS9Isw>.
- [255] Shengbang Tong, Erik Jones, and Jacob Steinhardt. Mass-producing failures of multimodal systems with language models, 2023.
- [256] Maya Grace Torii, Takahito Murakami, and Yoichi Ochiai. Expanding horizons in hci research through llm-driven qualitative analysis, 2024.
- [257] Antonio Torralba and Alexei Efros. Unbiased look at dataset bias. In *CVPR*, 2011.
- [258] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.
- [259] Nanne van Noord. Prototype-based dataset comparison. In *ICCV*, 2023.
- [260] Joshua Vendrow, Saachi Jain, Logan Engstrom, and Aleksander Madry. Dataset interfaces: Diagnosing model failures using controllable counterfactual generation. In *ArXiv preprint arXiv:2302.07865*, 2023.
- [261] Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. Replacing judges with juries: Evaluating llm generations with a panel of diverse models, 2024.
- [262] Pat Verga, Sebastian Hofstatter, Sophia Althammer, Yixuan Su, Aleksandra Piktus, Arkady Arkhangorodsky, Minjie Xu, Naomi White, and Patrick Lewis. Replacing judges with juries: Evaluating llm generations with a panel of diverse models. *arXiv preprint arXiv:2404.18796*, 2024.
- [263] Vijay Viswanathan, Kiril Gashteovski, Carolin Lawrence, Tongshuang Wu, and Graham Neubig. Large language models enable few-shot clustering, 2023. URL <https://arxiv.org/abs/2307.00524>.
- [264] Ulrike von Luxburg. A tutorial on spectral clustering, 2007. URL <https://arxiv.org/abs/0711.0189>.
- [265] Kiri Wagstaff, Claire Cardie, Seth Rogers, and Stefan Schrödl. Constrained k-means clustering with background knowledge. pages 577–584, 01 2001.
- [266] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. *arXiv preprint 1905.00537*, 2019.

- [267] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. 2019. In the Proceedings of ICLR.
- [268] Angelina Wang, Arvind Narayanan, and Olga Russakovsky. REVISE: A tool for measuring and mitigating bias in visual datasets. In *ECCV*, 2020.
- [269] Jialu Wang, Xinyue Gabby Liu, Zonglin Di, Yang Liu, and Xin Eric Wang. T2iat: Measuring valence and stereotypical biases in text-to-image generation, 2023.
- [270] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. Improving text embeddings with large language models, 2024. URL <https://arxiv.org/abs/2401.00368>.
- [271] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution, 2024. URL <https://arxiv.org/abs/2409.12191>.
- [272] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.754. URL <https://aclanthology.org/2023.acl-long.754>.
- [273] Zeyu Wang, Klint Qinami, Ioannis Christos Karakozis, Kyle Genova, Prem Nair, Kenji Hata, and Olga Russakovsky. Towards fairness in visual recognition: Effective strategies for bias mitigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. doi: 10.48550/arXiv.1911.11834. URL <https://arxiv.org/abs/1911.11834>.
- [274] Zijie J. Wang, Evan Montoya, David Munechika, Haoyang Yang, Benjamin Hoover, and Duen Horng Chau. DiffusionDB: A large-scale prompt gallery dataset for text-to-image generative models. *arXiv preprint arXiv:2210.14896*, 2022.
- [275] William Webber, Alistair Moffat, and Justin Zobel. A similarity measure for indefinite rankings. *ACM Trans. Inf. Syst.*, 28:20:1–20:38, 2010. URL <https://api.semanticscholar.org/CorpusID:16050561>.
- [276] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.

- [277] Eric Xing, Michael Jordan, Stuart J Russell, and Andrew Ng. Distance metric learning with application to clustering with side-information. In S. Becker, S. Thrun, and K. Obermayer, editors, *Advances in Neural Information Processing Systems*, volume 15. MIT Press, 2002. URL https://proceedings.neurips.cc/paper_files/paper/2002/file/c3e4035af2a1cde9f21e1ae1951ac80b-Paper.pdf.
- [278] Blair Yang, Fuyang Cui, Keiran Paster, Jimmy Ba, Pashootan Vaezipoor, Silviu Pitis, and Michael R. Zhang. Report cards: Qualitative evaluation of language models using natural language summaries, 2024. URL <https://arxiv.org/abs/2409.00844>.
- [279] Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E. Gonzalez, and Ion Stoica. Rethinking benchmark and contamination for language models with rephrased samples, 2023.
- [280] Linli Yao, Weiying Wang, and Qin Jin. Image difference captioning with pre-training and contrastive learning. In *AAAI*, 2022.
- [281] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *TMLR*, 2022.
- [282] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.
- [283] Xiang Yue, Yuansheng Ni, Kai Zhang, Tianyu Zheng, Ruoqi Liu, Ge Zhang, Samuel Stevens, Dongfu Jiang, Weiming Ren, Yuxuan Sun, Cong Wei, Botao Yu, Ruibin Yuan, Renliang Sun, Ming Yin, Boyuan Zheng, Zhenzhu Yang, Yibo Liu, Wenhao Huang, Huan Sun, Yu Su, and Wenhui Chen. Mmmu: A massive multi-discipline multimodal understanding and reasoning benchmark for expert agi, 2024. URL <https://arxiv.org/abs/2311.16502>.
- [284] Xiang Yue, Tianyu Zheng, Yuansheng Ni, Yubo Wang, Kai Zhang, Shengbang Tong, Yuxuan Sun, Botao Yu, Ge Zhang, Huan Sun, Yu Su, Wenhui Chen, and Graham Neubig. Mmmu-pro: A more robust multi-discipline multimodal understanding benchmark. *arXiv preprint arXiv:2409.02813*, 2024.
- [285] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, 2019.
- [286] Michael JQ Zhang, Zhilin Wang, Jena D. Hwang, Yi Dong, Olivier Delalleau, Yejin Choi, Eunsol Choi, Xiang Ren, and Valentina Pyatkin. Diverging preferences: When do annotators disagree and do models know?, 2024. URL <https://arxiv.org/abs/2410.14632>.
- [287] Peng Zhang, Yash Goyal, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Yin and Yang: Balancing and answering binary visual questions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

- [288] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.
- [289] Yian Zhang, Yifan Mai, Josselin Somerville Roberts, Rishi Bommasani, Yann Dubois, and Percy Liang. Helm instruct: A multidimensional instruction following evaluation framework with absolute ratings, February 2024. URL <https://crfm.stanford.edu/2024/02/18/helm-instruct.html>.
- [290] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric. P Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. Lmsys-chat-1m: A large-scale real-world llm conversation dataset, 2023.
- [291] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Tianle Li, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Zi Lin, Eric. P Xing, Joseph E. Gonzalez, Ion Stoica, and Hao Zhang. Lmsys-chat-1m: A large-scale real-world llm conversation dataset, 2023.
- [292] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.
- [293] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. *NeurIPS*, 2023.
- [294] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. In *NeurIPS Datasets and Benchmarks*, 2023.
- [295] Ruiqi Zhong, Charlie Snell, Dan Klein, and Jacob Steinhardt. Describing differences between text distributions with natural language, 2022. URL <https://arxiv.org/abs/2201.12323>.
- [296] Ruiqi Zhong, Charlie Snell, Dan Klein, and Jacob Steinhardt. Describing differences between text distributions with natural language. In *ICML*, 2022.
- [297] Ruiqi Zhong, Peter Zhang, Steve Li, Jinwoo Ahn, Dan Klein, and Jacob Steinhardt. Goal driven discovery of distributional differences via language descriptions, 2023. URL <https://arxiv.org/abs/2302.14233>.
- [298] Ruiqi Zhong, Peter Zhang, Steve Li, Jinwoo Ahn, Dan Klein, and Jacob Steinhardt. Goal driven discovery of distributional differences via language descriptions. *arXiv preprint arXiv:2302.14233*, 2023.
- [299] Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models. *arXiv preprint arXiv:2304.06364*, 2023.

- [300] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016.
- [301] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.
- [302] Banghua Zhu, Evan Frick, Tianhao Wu, Hanlin Zhu, and Jiantao Jiao. Starling-7b: Improving llm helpfulness & harmlessness with rlaiif, November 2023.
- [303] Jun-Yan Zhu, Yong Jae Lee, and Alexei A Efros. Averageexplorer: Interactive exploration and alignment of visual data collections. In *SIGGRAPH*, 2014.
- [304] Zhiying Zhu, Weixin Liang, and James Zou. Gsclip: A framework for explaining distribution shifts in natural language. In *ICML DataPerf Workshop*, 2022.
- [305] Lütü Kerem Şenel, Furkan Şahinüç, Veysel Yücesoy, Hinrich Schütze, Tolga Çukur, and Aykut Koç. Learning interpretable word embeddings via bidirectional alignment of dimensions with semantic concepts. *Information Processing & Management*, 59(3):102925, 2022. ISSN 0306-4573. doi: <https://doi.org/10.1016/j.ipm.2022.102925>. URL <https://www.sciencedirect.com/science/article/pii/S0306457322000498>.