UNIVERSITY OF CALIFORNIA SAN DIEGO

Personalizing Interactive Agents

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Shuyang Li

2022

University of California San Diego

2022

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I am sincerely grateful and deeply indebted to Professor Julian McAuley for supporting and guiding my research interests (even if some of them were a little off the beaten trail), and opening up opportunities for research and collaboration I would have never imagined before. I am grateful to my committee members, Professor Taylor Berg-Kirkpatrick, Professor Garrison Cottrell, Professor Virginia de Sa, and Professor Jingbo Shang for their encouragement and valuable feedback throughout these past years.

I am infinitely thankful for my parents, Daizong and Yu-Qing, and for their encouragement, nurturing, and unshakeable belief in me throughout these twenty-something years and counting. I have been so fortunate to have my wife Rei by my side through this process—your love and support has been invaluable to my growth and success.

I have had the honor and privilege of working with amazing colleagues and collaborators at UC San Diego, including Bodhisattwa Prasad Majumder, Jianmo Ni, Henry Mao, and Yufei Li, who have been a constant source of inspiration in this journey. I greatly appreciate my industry mentors who helped guide my research: Mukund Sridhar, Jin Cao, Chandana Satya Prakash, and Wael Hamza at Amazon; Sachin Ramachandran, Semih Yavuz, and Yingbo Zhou at Salesforce; and Jacky Wang at Google.

A very special thanks to Alicia Li, Micah Iticovici, Kathy Chen, Allen Chen, and all of my other wonderful friends in San Diego, New York, and across the globe for their unwavering support and companionship through years of work, school, a pandemic, and so much more. You have kept me sane and given me the motivation to forge ahead in my research.

The following includes per-chapter acknowledgements of material:

Chapter 2 contains material as it appears in Empirical Methods in Natural Language Processing, 2020 (“Interview: Large-scale Modeling of Media Dialog with Discourse Patterns and Knowledge Grounding” Bodhisattwa Prasad Majumder*, Shuyang Li*, Jianmo Ni, Julian McAuley). The dissertation author was the co-primary investigator and author of this paper.

Chapter 3, in part contains material as it appears in Conference of the European Chapter of the Association for Computational Linguistics, 2021 (“Zero-shot Generalization in Dialog State Tracking through Generative Question Answering” Shuyang Li, Jin Cao, Mukund Sridhar, Henghui Zhu, Shang-Wen Li, Wael Hamza, Julian McAuley). The dissertation author was the primary investigator and author of this paper.

Chapter 3, in part contains material as it appears in Findings of Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2022 (“Instilling Type Knowledge in Language Models via Multi-Task QA” Shuyang Li, Mukund Sridhar, Chandana Satya Prakash, Jin Cao, Wael Hamza, Julian McAuley). The dissertation author was the primary investigator and author of this paper.

Chapter 4, contains material as it appears in the Second Workshop on Trustworthy Natural Language Processing at NAACL 2022 (“Self-Supervised Bot Play for Transcript-Free Conversational Recommendation with Rationales” Shuyang Li, Bodhisattwa Prasad Majumder, Julian McAuley). The dissertation author was the primary investigator and author of this paper.

Chapter 5 contains material as it appears in Empirical Methods in Natural Language Processing, 2019 (“Generating Personalized Recipes from Historical User Preferences” Bodhisattwa Prasad Majumder*, Shuyang Li*, Jianmo Ni, Julian McAuley). The dissertation author was the co-primary investigator and author of this paper.

Chapter 6, in part is currently being prepared for submission for publication of the material “SHARE: a System for Hierarchical Assistive Recipe Editing” Shuyang Li, Yufei Li, Jianmo Ni, Julian McAuley. The dissertation author was the primary investigator and author of this paper.

VITA

2016	B.S.E., Princeton University
2021	M.S., University of California San Diego
2022	Ph.D., University of California San Diego

PUBLICATIONS

Shuyang Li, Bodhisattwa Prasad Majumder, Julian McAuley. "Self-Supervised Bot Play for Transcript-Free Conversational Recommendation with Rationales". *Second Workshop on Trustworthy Natural Language Processing* (TrustNLP at NAACL). 2022.

Shuyang Li, Mukund Sridhar, Chandana Satya Prakash, Jin Cao, Wael Hamza, Julian McAuley. "Instilling Type Knowledge in Language Models via Multi-Task QA". *Findings of Annual Conference of the North American Chapter of the Association for Computational Linguistics* (Findings of NAACL). 2022.

Shuyang Li, Jin Cao, Mukund Sridhar, Henghui Zhu, Shang-Wen Li, Wael Hamza, Julian McAuley. "Zero-shot Generalization in Dialog State Tracking through Generative Question Answering". *Conference of the European Chapter of the Association for Computational Linguistics* (EACL). 2021.

Bodhisattwa Prasad Majumder*, Shuyang Li*, Jianmo Ni, Julian McAuley. "Interview: Large-scale Modeling of Media Dialog with Discourse Patterns and Knowledge Grounding". *Empirical Methods in Natural Language Processing* (EMNLP). 2020.

Henry Huanru Mao, Shuyang Li, Julian McAuley, Garrison Cottrell. "Speech Recognition and Multi-Speaker Diarization of Long Conversations". *INTERSPEECH*. 2020.

Bodhisattwa Prasad Majumder*, Shuyang Li*, Jianmo Ni, Julian McAuley. "Generating Personalized Recipes from Historical User Preferences". *Empirical Methods in Natural Language Processing* (EMNLP). 2019.

Dongyue Huang, Shuyang Li, Ajay Dhaka, Gina M. Story, Yu-Qing Cao. "Expression of the Transient Receptor Potential Channels TRPV1, TRPA1 and TRPM8 in Mouse Trigeminal Primary Afferent Neurons Innervating the Dura". *Molecular Pain*. 2012.

ABSTRACT OF THE DISSERTATION

Personalizing Interactive Agents

by

Shuyang Li

Doctor of Philosophy in Computer Science

University of California San Diego, 2022

Professor Julian McAuley, Chair

The recent proliferation of machine learning (ML) agents that interact with humans through natural text conversation (e.g. smart phone assistants, chat bots) is predicated on the unprecedented public availability of user-contributed text (e.g. blogs, product reviews) and behavioral traces (e.g. purchases, social media interactions). Current methods for building conversational agents have seen success in highly structured fields like automated help desks and reservation booking. However, it remains challenging to apply these ML systems to help users with daily tasks in more natural and intuitive ways. For example, current recommender systems cannot fluidly engage with users for multiple rounds of conversation.

In this dissertation we focus primarily on developing technologies that allow intelligent agents to engage with users in trustworthy, personalized, and interactive ways through the medium of text. Specifically, my work focuses on 1) explainable dialog models to facilitate meaningful interviews; 2) a language modeling framework to infer user preferences from dialog; and 3) a bot-play framework for training explainable and personalized recommender systems to understand and reflect user feedback over multiple turns of conversation. I finally present two case studies on applying the aforementioned technologies to build personalized interactive agents that generate and edit instructional texts (e.g. cooking recipes) to assist users in their daily lives.

Chapter 1

Introduction

One longstanding goal of artificial intelligence is the development of intelligent agents that can understand human intents and preferences, and can interact with users accordingly. We aim to build such agents that are *personalized*—can provide services and experiences tailored to an individual user—and *interactive*—can engage with users and change their behavior based on human feedback. We build upon extensive work in machine learning that seeks to understand human behavior and interface with humans via natural text (e.g. spoken dialog, chat logs, product reviews). Advances in personalized interactive agents can lead to improvements in a wide range of consumer-focused applications including chatbots, home assistants, customer service, health and dietary advice, and entertainment media.

The enormous amount of textual data (e.g. dialog transcripts, blogs, product reviews) and behavioral data (e.g. website purchase records, product ratings) available and continuously updated to the internet has revealed opportunities and challenges for the study and development of personalized interactive agents. As online services and platforms cover more diverse and specific domains, the need for personalization has become ever more relevant. Information and products that are relevant for one user may not serve another user well, particularly for highly individual preferences like diet and entertainment. Meanwhile, many user-facing applications such as product recommendations and cooking recipes are static, affording users no way to tailor the experience to their own needs and wants.

We can build personalized interactive agents in two ways: 1) adapting interactive agents to respond to feedback in ways specifically tailored to each user; and 2) adapting personalized models (e.g. recommender systems) to accommodate user feedback over multiple rounds of engagement. Here we focus on interactions in the modality of natural text: models that can understand and generate text (e.g. a free-text response to a user, or a structured text output like an instruction set). We have identified three key challenges for personalizing interactive agents:

1. To be *trustworthy*, the agent must be able to *explain* its suggestions or outputs to the user.
2. To be *personalized*, the agent must accommodate natural-text feedback from a user and from that feedback be able to infer the user's preferences.
3. To be *interactive*, the agent must be able to incorporate user feedback into an updated response (e.g. new suggestions, edited recipes or documents).

1.1 Methods for Personalizing Interactive Agents

In the first part of this dissertation, we present methods to instill *trust*, *personalization*, and *interactivity* in assistive agents. We have studied how to build explainable models for generating text, proposed frameworks to understand and infer user preferences from free-form dialog, and developed techniques to train interactive models with easily accessible corpora.

1.1.1 Explainable Models for Text Generation

With the advent of widely-available chat agents like Amazon Alexa [60] and Microsoft Cortana [190], humans increasingly interact with machine learning models via the modality of text.[1] However, these assistive agents often rely on powerful language models that encode input text (e.g. user utterances) into a latent vector space which is then used to generate a response [13, 77, 236]. These learned methods are black-box, with little to no opportunity for users to discover why certain decisions or predictions are made.

[1]While many AI assistants vocalize their responses, typically the model outputs predictions in the form of text, which is then passed through a text-to-speech service [81].

Meanwhile, in two-party conversations between humans, the success and perceived quality of dialog depends heavily on how much each participant understands and trusts each other's reasoning. This fundamental principle of human interactions has been observed across varied domains, including customer service [34], education [33], and healthcare [40, 69]. The importance of explainable reasoning is also tied to the successful adoption of product recommendation services [164, 193] and computer interfaces [22].

In order to build more explainable models for text generation, my research focuses on models that can ground their predictions on user behavioral data and external knowledge, allowing us to clearly see what information a model relies on to generate responses. I have studied methods to associate conversations with real-world events in order to generate responses that are topical, informative, and specific. I specifically study how to model discourse patterns and usage of external knowledge when asking questions in news interviews. We develop a simple yet effective method to associate relevant news headlines to media interviews. By training our models to explain which news articles are most relevant to a conversation, we can generate more fluent and informative responses compared to strong baselines.

1.1.2 Extracting User Preferences from Dialog

Useful interactive agents must not only be capable of explaining the reason behind their suggestions—they must also understand user preferences as conveyed through natural dialog. Users can interact with a variety of chat-bot agents via free text, including for chit-chat [137, 176], hotel and restaurant booking [13], or product recommendation [113]. While some dialog systems encode user utterances to directly generate responses [83, 113], users often express explicit preferences (e.g. a specific cuisine when looking for a restaurant, an actor's name when looking for a movie, or a color of fabric when buying clothes) in their statements. Extracting such preferences is particularly important for *task-oriented* or *goal-oriented* dialog systems, including services offered by Google Assistant [128] or Apple Siri [5].

Preference extraction is often posed as *dialog state tracking* (DST)—a slot-filling problem

[106]: given a domain or service (e.g. restaurant booking), we aim to infer the user's preferences about a set of attributes (e.g. cuisine or neighborhood). This paradigm is easily adopted for domains and services with plentiful training data, but cannot be readily applied to new services or arbitrary new domains. In particular, assistants that are capable of open-domain chit-chat (e.g. to identify which services a user is most likely to need, or to provide general help to users) can interact with users wanting to talk about developing events and timely knowledge. In this *zero-shot* setting, we aim to develop an agent that can infer specific user preferences for arbitrary new conversations and previously unseen entities [20]. Models capable of zero-shot DST can be more useful to users who have specific or rare preferences, such as when recommending products in a complex space (e.g. books, music) or adding agent capabilities/services.

To address this challenge, my research focuses on using generative modalities to enable dialog models to infer user preferences in conversations about new domains. I introduce an ontology-free framework that supports natural language queries for unseen constraints and slots in multi-domain task-oriented dialog. This allows a service or developer to discover fine-grained preferences from new conversational topics without costly model re-training. Our generative question-answering approach improves zero-shot joint goal accuracy by up to 9 points compared to previous state-of-the-art (SOTA).

I also study how user preferences can be mapped to *types*—an indication of where an entity belongs in our taxonomy of knowledge (e.g. Chinese food has a type *cuisine*). By teaching models to recognize type knowledge in a broad set of informative documents, we aim for these models to better recognize new types and topics expressed in dialog. I propose a multi-task question-answering framework to allow language models to understand and recognize entities and their types. We demonstrate that our framework allows language models to more accurately recognize types and concepts found in informational text, as well as infer novel, highly specific types for unseen entities. We further apply models trained using this framework to our aforementioned question-answering formulation for zero-shot DST, achieving additional improvements in zero-shot joint goal accuracy by up to 14.9 points.

1.1.3 Enabling Multi-Turn Interactive Agents

Once we have a machine assistant that can articulate its reasoning and understand preferences expressed by the user, the final piece of the puzzle lies in using those preferences to make new suggestions that better fit the user's goals. This mirrors a common paradigm of human interaction in daily life: when trying to accomplish a task (e.g. deciding what to eat, or buying a product), people regularly refine their preferences and change their advice based on evolving requirements [164, 249]. For example, when deciding which restaurant to dine at, people commonly add requirements as they consider suggestions:

- **Person A:** I'm hungry. Do you know anything good around here?
- **Person B:** There are a few nice Korean BBQ restaurants nearby.
- **A:** I'm not feeling Asian, what about something American?
- **B:** We could check out the new diner that opened up down the block.
- **A:** I kind of want something fancier.
- **B:** What about a steakhouse?
- **A:** Great—let's go!

When designing a chat assistant to mimic Person B (the recommender), we must be able to change our suggestions based on subjective feedback (e.g. *I kind of want something fancier.*). Previous work in conversational agents generally fall under three bins. First are interactive search agents which ask the user questions about specific attributes of products (e.g. *Do you want a **blue** dress?*) in order to narrow the search space of relevant products [12, 104, 247, 105]. These models often limit the user's agency, requiring them to give binary feedback on the queried attribute.

On the other hand, dialog agents interact with the user in the form of natural language, allowing users to freely give feedback [113, 83]. Dialog agents learn to respond to users by training over carefully curated, human-annotated corpora of domain-specific conversation transcripts. As a result, it can be prohibitively expensive to collect the data to train such models for complex domains where users have specific preferences (e.g. electronics purchases, alcohol recommendations, or health advice).

Finally, conversational critiquing systems allow users to interact with a wide range of possible attributes at each turn, incrementally constructing user profiles. However, current critiquing systems can only be trained for next-item recommendation [227, 133, 4] or a single turn of critiquing [3]. These models thus struggle to incorporate user feedback in multi-turn settings like our example above.

We address the downsides of these three existing approaches by proposing a simple yet effective bot-play framework to train multi-turn conversational recommender systems. Our framework uses review data to synthesize and model user feedback over multiple turns of conversation, allowing us to train conversational recommenders on any domain with user-written reviews—without the need for expensive transcript collection. We demonstrate that our method is model-agnostic and allows simple matrix factorization and linear recommender systems to out-perform SOTA techniques for conversational recommendation.

1.2 Case Studies: Personalized Interactive Agents for Instructional Texts

In the second part of this dissertation, we present two case studies on applying our techniques to build *trustworthy*, *personalized*, and *interactive* assistants to generate instructional texts for daily use. We specifically focus on generating cooking recipes: tools that can help millions of home cooks discover dishes that fit their culinary preferences while reflecting their dietary requirements.

First, I propose a method to generate instructions given some known ingredients and a user's previously consumed recipes. While previous methods for generating recipes would return the same recipe for each set of ingredients [92, 102], cooks are more likely to cook dishes that fit their individual tastes. Our model allows users to, for example, find something new and tasty to cook with the ingredients they have in their refrigerator. We compute attention scores over the user's previously consumed recipes to guide the instruction generation. These scores also allow the model to explain to the user from where each instruction takes its inspiration.

Next, I build a system to *edit* recipes to assist home cooks with dietary restrictions. As we are mainly concerned with agents that can interact with humans via natural language, we build on previous work in controllable text generation [174]. While prior techniques for controllable generation focus on stylistic attributes of text including politeness [135, 153], formality [234], and persona [111, 141], we seek ways to control the substance of generated text based on complex preferences. For example, cooks and diners with dietary restrictions are poorly-served by existing online resources. We aim to edit generated recipes to satisfy the complex requirements—that all ingredients are safe for the user, and the recipe is structurally sound (can be followed to create real food). We show that our system can create new versions of recipes that satisfy dietary constraints and can be successfully cooked.

1.3 Thesis Organization

This thesis is organized into two parts and seven chapters. Part 1 contains chapters 2, 3, and 4, which comprise our work in a) building explainable models for natural text generation, b) extracting preferences from natural-text user utterances in dialog, and c) teaching recommender systems to incorporate user feedback. Part 2 contains two case studies—chapters 5 and 6—on applying our work to assistive agents for daily life, specifically for generating novel instructional texts (cooking recipes) to satisfy user preferences and specific constraints. Finally, we conclude the dissertation and discuss future topics of research in Chapter 7.

We provide a brief overview of the main contributions in each chapter as follows:

Modeling Media Discourse with Knowledge Grounding: In this chapter, we perform the first large-scale analysis of discourse in media dialog and its impact on generative modeling of dialog turns, with a focus on interrogative patterns and use of external knowledge. Discourse analysis can help us understand modes of persuasion, entertainment, and information elicitation in such settings, but has been limited to manual review of small corpora. We introduce INTERVIEW—a large-scale (105K conversations) media dialog dataset collected from news interview transcripts—which allows us to investigate such patterns at scale. We build a dialog model that learns to align relevant external knowledge to interviews and predict discourse patterns. We demonstrate that our model quantitatively and qualitatively outperforms strong discourse-agnostic baselines for dialog modeling—generating more specific and topical responses in interview-style conversations.

Type-Aware Question-Answering for Zero-shot Dialog State Tracking: In this chapter, we study Dialog State Tracking (DST): an integral part of modern dialog systems that aims to track user preferences and constraints (slots) in task-oriented dialogs. In real-world settings with constantly changing services, DST systems must generalize to new domains and unseen slot types. We introduce a novel ontology-free generative question-answering framework that supports natural language queries for unseen constraints and slots in multi-domain task-oriented dialogs. Our model improves joint goal accuracy (JGA) in zero-shot domain adaptation settings by up to 9% (absolute) over the previous state-of-the-art on the MultiWOZ 2.1 dataset. We further introduce a method to instill fine-grained type knowledge—identifying entities and their place in a taxonomy of knowledge—in language models with text-to-text pre-training on type-centric questions leveraging knowledge base documents and knowledge graphs. We create the **WikiWiki** dataset: entities and passages from 10M Wikipedia articles linked to the Wikidata knowledge graph with 41K types. Models trained on WikiWiki achieve further SOTA performance in zero-shot DST benchmarks, accurately infer entity types in Wikipedia articles, and can discover new types deemed useful by human judges.

Transcript-Free Conversational Recommendation with Rationales: In this chapter, we study conversational recommender systems that offer a way for users to engage in multi-turn conversations to find items they enjoy. For users to trust an agent and give effective feedback, the recommender system must be able to *explain* its suggestions and rationales. We develop a two-part framework for training multi-turn conversational recommenders that provide recommendation rationales that users can effectively interact with to receive better recommendations. First, we train a recommender system to jointly suggest items and explain its reasoning via subjective rationales. We then fine-tune this model to incorporate iterative user feedback via self-supervised bot-play. Experiments on three real-world datasets demonstrate that our system can be applied to different recommendation models across diverse domains to achieve state-of-the-art performance in multi-turn recommendation. Human studies show that systems trained with our framework provide more useful, helpful, and knowledgeable suggestions in warm- and cold-start settings. Our framework allows us to use only product reviews during training, avoiding the need for expensive dialog transcript datasets that limit the applicability of previous conversational recommender agents.

Personalized Cooking Recipe Generation from Historical Interactions: In this chapter, we consider the novel problem of *personalized recipe generation* to assist users with culinary preferences but incomplete knowledge of ingredients for a specific dish. We aim to expand a name and incomplete ingredient details into complete natural-text instructions aligned with the user's historical preferences. We attend on technique- and recipe-level representations of a user's previously consumed recipes, fusing these 'user-aware' representations in an attention fusion layer to control recipe text generation. Experiments on a new dataset of 180K recipes and 700K interactions show our model's ability to generate plausible and personalized recipes compared to non-personalized baselines.

Assistive Recipe Editing: In this chapter, we seek to help home cooks with dietary restrictions by tackling the challenge of *controllable recipe editing*: adapting a base recipe to satisfy a user-specified dietary constraint. This task is challenging, and cannot be adequately solved with human-written ingredient substitution rules or existing end-to-end recipe generation models. We tackle this problem with SHARE: a **S**ystem for **H**ierarchical **A**ssistive **R**ecipe

Editing, which performs simultaneous ingredient substitution before generating natural-language steps using the edited ingredients. By decoupling ingredient and step editing, our step generator can explicitly integrate the available ingredients. Experiments on the novel *RecipePairs* dataset demonstrate that SHARE produces convincing, coherent recipes that are appropriate for a target dietary constraint. We further show through human evaluations and real-world cooking trials that recipes edited by SHARE can be easily followed by home cooks to create appealing dishes.

Part I

Methods for Personalizing Interactive Agents

Chapter 2

Modeling Media Discourse with Knowledge Grounding

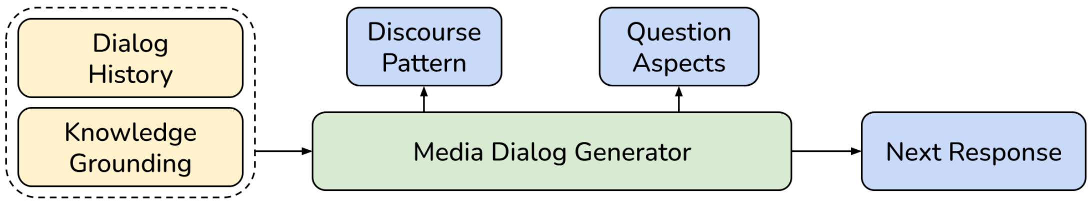


Figure 2.1. Our dialog model incorporates grounding documents alongside dialog history. We also leverage the dialog patterns and interrogative positioning by the host via auxiliary losses.

2.1 Introduction

Much of the news, information, and punditry the general public listens to and reads consists of *media dialog*—a category of open-domain conversations between an interviewer and interviewee centered on world events and situational context. A system for modeling media dialog from the perspective of one of these roles can help us better understand how media persuades and informs the public [194]. While recent work in dialog modeling has focused on goal-oriented [13], spontaneous [188], or synthetic open-domain chit-chat [118, 39, 60], there is a paucity of large-scale analysis of discourse and knowledge grounding in media dialog. **In this chapter,** we aim to build a framework for modeling media dialog in a way that explains which news articles and events are most relevant to the conversation and predicts discourse patterns over future turns.

Media dialog differs linguistically and in purpose from unstructured, spontaneous conversation such as open-domain chitchat, and both the topical content and interlocutor intent are heavily influenced by the social, cultural, and temporal setting [221]. The study of media dialog has traditionally focused on individual and manual review of small-scale (<200K word) news corpora [10, 208], and we see an opportunity to scale some forms of discourse analysis to tens of thousands of such documents. In this chapter, we perform the first large-scale automatic analysis of structural components (response-type patterns) and question type categorization on media dialog, specifically for English news interviews.

We show that predicting discourse features can improve generative dialog modeling performance, demonstrating the degree to which discourse structure impacts an interviewer's choice of response type and content. News interviews are also heavily situation-grounded and contextualized by past events and world knowledge. We explore methods to associate each conversation with a selection of world facts, and show that by modeling interviewers as *knowledge-grounded* speakers mediating a conversation we are able to generate relevant and specific utterances fitting their role.

Our main contributions in this chapter are:

1. We collect a dataset of 105K media dialogs (23K two-party dialogs)[1] encompassing two decades of National Public Radio (NPR) radio programs, on which we conduct extensive experiments;

2. We present a probabilistic framework to link a dialog with facts from a large corpus of grounding documents and show that it improves downstream dialog modeling performance compared to a strong TF-IDF baseline;

3. We introduce two auxiliary losses to guide utterance generation in a media dialog setting: look-ahead dialog structure prediction and question-attribute prediction. We show that these losses significantly improve generation quality via automatic and human metrics.

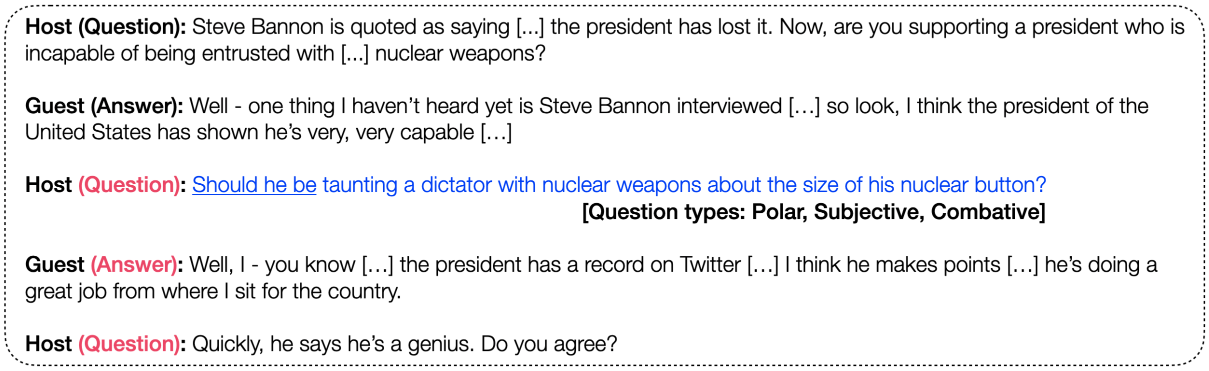
[1]https://www.kaggle.com/shuyangli94/interview-npr-media-dialog-transcripts

2.2 Related Work

Media dialog—specifically, the news interview—has seen study primarily in the field of speech transcription, diarization, and speaker role modeling [24, 101]. These works have typically focused on techniques to annotate broadcast audio transcripts [80] in order to cluster different news stories from a continuous broadcast stream [79]. While Barzilay et al. [9] and Liu [125] note that transition points between speaker roles (e.g. anchor and guest) can determine the high-level topical flow of a news conversation, we investigate the impact of discourse patterns on the semantics of specific utterances.

Such research is currently limited by a lack of accessible corpora for the study of media dialog at scale. The Defense Advanced Research Projects Agency (DARPA) has undertaken efforts to collect and transcribe broadcast conversations [196, 32]. However, it proves difficult to adopt these datasets as widely available benchmarks on dialog modeling tasks, as they come with a substantial cost ($100-$1000 per annum per dataset). More recent efforts to amass such data have either focused on collecting large volumes of conversation fragments with noisy transcripts [11] or human transcripts for a smaller set of long-form open-domain radio programs [139]. We contribute an open-access large-scale corpus of broadcast media dialog annotated with response types, demonstrating that these are useful for modeling interviewer utterances.

We explore the application of discourse analysis [45] on this large media dialog corpus in order to discover, confirm, and leverage *discourse patterns* regarding interrogative forms, speaker agency, and references to external knowledge. As noted by Weizman [221] in their deep study of Israeli news television, structure in media dialog (in contrast to spontaneous natural conversation) is uniquely determined by its speaker role dynamics. Wang et al. [217] investigate the detection of one such dynamic: agreement/disagreement between speakers. Ma et al. [134] classify discourse relations (e.g. comparative, temporal) between two turns of dialog, but do not study discourse structure. In this chapter we extend our analysis to other properties of interviewer utterances (e.g. subjectivity, polarity, dialog act patterns) [73] in the context of generative


Host (Question): Steve Bannon is quoted as saying [...] the president has lost it. Now, are you supporting a president who is incapable of being entrusted with [...] nuclear weapons?

Guest (Answer): Well - one thing I haven't heard yet is Steve Bannon interviewed [...] so look, I think the president of the United States has shown he's very, very capable [...]

Host (Question): Should he be taunting a dictator with nuclear weapons about the size of his nuclear button?
[Question types: Polar, Subjective, Combative]

Guest (Answer): Well, I - you know [...] the president has a record on Twitter [...] I think he makes points [...] he's doing a great job from where I sit for the country.

Host (Question): Quickly, he says he's a genius. Do you agree?

Figure 2.2. Example conversation from INTERVIEW with annotated discourse analysis. Text highlighted in blue indicates the question of interest, uttered by the host. The dialog triplet is marked in red.

dialog modeling. Structured approaches for dialog modeling employ a simple concatenation of dialog history in a transformer-based architecture [245]. We draw inspiration from Luan et al. [130] who demonstrate the usefulness of a multi-task framework for speaker-conditioned dialog modeling. Guu et al. [66] propose a framework for jointly learning document retrieval and language modeling, and we propose a similar model to learn task-specific annotation of grounding documents.

2.3 INTERVIEW: A Media Dialog Corpus

We collect a new dataset of 105K multi-party interview transcripts for 7 programs on National Public Radio (NPR)[2] over 20 years (1999–2019). These transcripts contain in total 3M turns comprising 7.5M sentences (127M words) from 184K speakers, of which 287 are interviewers. To investigate host-mediated media dialog, we curate a subset, **INTERVIEW 2P**, with **two roles**: an *interviewer* and a *guest*, comprising 23K two-party conversations encompassing 455K turns, with 1.24M sentences and 21.7M words. In these two-party conversations, each speaker takes an average of nine turns per dialog. Guests tend to speak longer on their turns, with 1.6x as many sentences spoken and 2x as many words per turn. Meanwhile, hosts ask five times as many questions as guests, with 40% of their dialog turns containing questions. When asking

[2]https://www.npr.org/

Table 2.1. Comparative media dialog dataset statistics, including the two-party (2P) and full INTERVIEW dataset. Note that RadioTalk does not contain full conversations

Dataset	Year	Structured	# Dialogs	# Turns	# Words
RadioTalk [11]	2019	✗	*5.98 M*	116 M	2.9 B
TAL [139]	2020	✓	663	163,808	7.4 M
INTERVIEW 2P	2020	✓	23,714	454,739	21.7 M
INTERVIEW	2020	✓	105,848	3,199,856	126.7 M

questions, hosts and guests use interrogative forms [184] at the same rate (65%).

We avoid modeling salutations and sign-offs (which tend to be formulaic, and specific to the radio station) by restricting the target turns to those with at least three prior turns and two following turns of conversation, resulting in a target training set of 87K host-only turns and 11K host-only turns for dev and test.

2.3.1 Comparison with Other Datasets

Open-domain dialog datasets have traditionally focused on either spontaneous (e.g. telephone calls) or goal-oriented conversation, and there is a paucity of English-language *media dialog* datasets—that is, dialog corpora comprising semi-structured conversations for the purpose of information elicitation and presentation. The closest such datasets are This American Life [139], a dataset of several hundred long-form expository podcast episodes, and RadioTalk [11], which comprises over one million ten-minute snippets of talk radio transcripts. While these corpora are derived from broadcast media, episodes of the former contain a broad range of expository speakers who are not professional journalists, while the latter dataset is constructed via an automated transcription system with a 13%+ word error rate and does not contain full conversations (segments from radio conversations are transcribed). We compare INTERVIEW statistics to other English media dialog datasets in Table 2.1.

Traditional media dialogs (e.g. news interviews) comprise a significant body of media consumed by the general public and we believe there is value in the large-scale study of such media. Efforts to collect and transcribe broadcast news span the world, from the French EPAC corpus [43] to Arabic and Chinese news manually transcribed via the GALE program [32]. To our knowledge, no attempt has yet been made to analyze the discourse patterns or trends in such data—these datasets have primarily been used to support the development of automatic speech recognition, transcription, and machine translation systems. Early efforts to collect English-language broadcast conversation transcripts [162] similarly aimed to build smaller, high-quality parallel corpora for speech transcription. The large-scale study of discourse in media dialog is not supported in such corpora, and the INTERVIEW corpus enables such analysis at scale for English-language media.

2.4 INTERVIEW Discourse Analysis

We tackle three aspects of discourse analysis that can be scaled to INTERVIEW: 1) Dialog patterns that emerge through new interviews; 2) Large scale annotation of interviewer question types (dialog acts); and 3) Obtaining grounding documents that provide situational context for an interview. We study these features in context of English broadcast news interviews.

2.4.1 Dialog Patterns

The news interview setting revolves around sets of questions and answers—naively, one may assume the interviewer to be the sole questioner. However, media dialog has steadily deviated from this rigid structure, tending toward the broadly conversational [44]. Each participant may be at turns jovial, inquisitive, and critical, and this is reflected in question-answer patterning. Heritage [73] frames the analysis of media discourse in terms of the *third-turn receipt*, where 1) they ask a question; 2) the interviewee responds; and 3) the interviewer chooses how to proceed. We are motivated by this, as well as studies of *question-response-confirmation* patterns in spontaneous dialog [209]. We focus on discourse patterns in **response type triplets** beginning with an interviewer (host) question.

We define a triplet as $\{r_1, r_2, r_3\}$ where the response type at utterance i is a question or an answer: $r_i \in \{\text{Q}, \text{A}\}$. By imposing a binary label on each utterance, we are able to efficiently mine all occurrences of each of eight possible host-guest-host patterns across our 23K dialogs. We find that a structured interrogative Q-A-Q pattern comprises 27% of all cases, while 20% of the time the host poses a non-interrogative third response (Q-A-A). Guests respond to questions with questions of their own only 7% of the time, supporting the theory that interviewers serve as the primary *mediators* in such conversations [221]. Manual inspection evinces recurring action patterns corresponding to interviewer stance-taking and agendas ranging from cooperative to confrontational. For example, the conversation segment in Figure 2.2 is comprised entirely of Q-A-Q patterns, with the host prompting [73] the guest, re-contextualizing and refocusing the guest's stance for the benefit of the audience. To leverage the inter-dependence of action choice (question or answer) and stance-taking (implicitly or explicitly via utterance content) [67], we propose to predict the subsequent response type triplet while modeling an interviewer utterance. We thus explore how utterance phrasing and structure may depend on projected or desired conversation directions.

2.4.2 Question Types as Dialog Acts

In their role as a mediator, interviewers can shape the narrative by posing different *types* of questions to guests. Weizman [221] posits that this choice of question type is influenced by dialog context and conversation flow. We examine ways to structurally bias our model to take advantage of conversational context in order to ask appropriate interviewer questions. Based on common interviewing guides[3] and linguistic analysis of open-ended questions in a conversational setting [86], we define three interrogative aspects (attributes): 1) **Polarity**: determining if the question is yes/no (polar) or open-ended; 2) **Subjectivity**: determining if it demands a factual answer or invites a subjective opinion; and 3) **Combativeness**: whether the question is confrontational or clarifying. Our mode of categorization resembles that of Gnisci and Bonaiuto [58], who add additional categories that are more relevant to the study of equivocation in confrontational interviews. While previous works have primarily used question polarity and interrogative forms

[3]http://prndg.org/host-interviewing-tips

Table 2.2. F1 score of question-type classifier models on the test set for different base classifier architectures, using single-utterance (left) and full dialog history (right) as model inputs.

	Single Utterance				With Dialog History			
	MLP	CNN	LSTM	BERT	MLP	CNN	LSTM	BERT
Polarity	55.61	68.20	66.87	75.31	68.71	74.71	70.49	80.20
Combativeness	48.91	57.19	49.70	58.10	60.81	65.87	60.54	70.14
Subjectivity	50.87	53.91	51.96	66.92	61.21	67.98	63.09	76.92

to improve diversity in spontaneous dialog generation [248], we explore how a news interviewer constructs question contents given desired interrogative aspects.

We hired two expert annotators to assess a question based on these three aspects. We provided interviewer questions alongside corresponding dialog histories, and annotators marked the binary presence/absence of each aspect for each question. The first host question from Figure 2.2—*Should [Trump] be taunting a dictator with nuclear weapons about the size of his nuclear button?*—would be marked as polar, subjective, and combative, as it asks the guest whether (polar) they endorse (subjective) an intentionally ridiculous statement (combative). We collected 1,000 questions in this manner, each labeled by both annotators. The inter-annotator agreement (Cohen's kappa [31]) for each of the binary labeling tasks—polar vs. open-ended, subjective vs. objective, combative vs. clarifying—was 0.8 for polarity, 0.72 for subjectivity and 0.7 for combativeness. We observed questions in this sample to be 60.2% polar, 38.7% subjective, and 29.5% combative.

Automatic Classification

We label the remainder of INTERVIEW by training a multi-label classifier, fine-tuning BERT [38] to predict the presence of each attribute in our human-annotated set of questions. We concatenate dialog history and the interviewer question separated by a `[SEP]` token and prepend a `[CLS]` token. We calculate binary cross entropy loss over a linear projection of the final hidden state of the `[CLS]` token. BERT achieves 80.20, 70.14, and 76.92 F1 scores for polarity, combativeness and subjectivity respectively on the test set in four epochs.

Algorithm 1: Pseudocode for probabilistic linking procedure initialized from TF-IDF.

```
Initialize document assignments from TF-IDF priors;
while Average validation perplexity decreases do
    Learning: Update the model with current assignments for $N$ epochs;
    for each $d$ in Dialogs do
        Sample $K$ documents from top 50 TF-IDF priors;
        for each $k$ in $K$ do
            Condition each response in the dialog with $k$, and calculate perplexity;
            Aggregate at the dialog level;
        Choose $k$ that yields the lowest perplexity;
    Assignment: Gather all $k$'s for each dialog to update current assignments;
```

We consider multiple baselines: 1) an MLP model using Bag-of-Words input features; 2) a CNN [52] with 2 convolution layers; and 3) a Bi-LSTM [63] network with max-pooling of final hidden layers. We initialize all embeddings with BERT embedding vectors. As shown in Table 2.2, BERT achieves the highest F1-score. Including dialog history improves classification performance, confirming that the type of question asked depends on conversational context. This suggests that we may also be able to better predict question content through jointly leveraging the dialog history and question type. Both human annotators and our model find predicting polarity the easiest, and combativeness the most difficult.

2.4.3 Knowledge Grounding

Media dialog is frequently characterized by references to world knowledge, current events, and factual information. This can be learned to some extent in large language models pre-trained on diverse text corpora [161], and such models can act as knowledge stores [27]. However, for tasks involving complex reasoning and induction it remains beneficial to provide models with externally linked knowledge [146, 47]. Specifically for dialog modeling, the Wizard of Wikipedia [39] and Topical Chat [60] corpora consist of grounding documents linked with open-domain chit-chat. As such, we explore methods to link *grounding knowledge documents* for each conversation in INTERVIEW, drawn from NPR news articles from the past two decades. We aim to link documents that can best inform conversation content and structure as measured

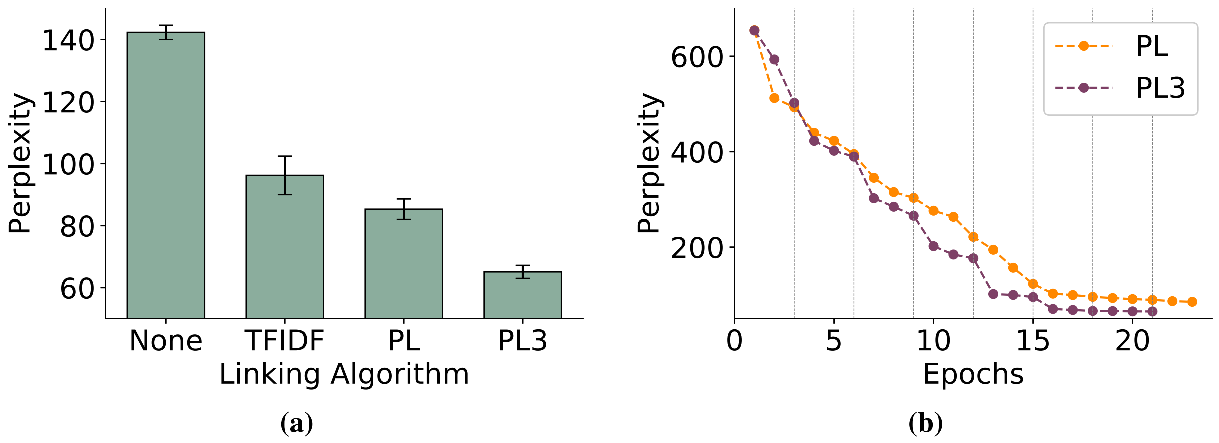


Figure 2.3. (a) Bar plot depicts test perplexity for linking algorithms: None (no grounding), TF-IDF, and PL/PL3 which indicate probabilistic linking with re-assignment at every 1/3 epochs respectively. Validation perplexity by epoch (b) shows that PL3 converges faster and to a better optimal.

by downstream dialog modeling performance.

TF-IDF Linking

We assess a strong retrieval baseline for grounding document linking, using TF-IDF [178] to find relevant documents for each conversation. To support large-scale TF-IDF similarity computation, we use the Lucene-based ElasticSearch [62] engine[4] to calculate TF-IDF similarity between full interview texts and the concatenation of the document headline and body, returning the 50 most similar grounding documents for each INTERVIEW conversation. We aim to link documents that would be reasonably relied on by the speakers at the time of the interview, and as such for each interview exclude articles that were published after the interview itself.

Probabilistic Linking

While TF-IDF based document linking provides a co-occurence-based similarity measure between documents and conversations, it is not guaranteed to improve dialog modeling performance. Thus, we train a linking model such that conditioning on linked documents has a positive effect on dialog modeling performance. We use a two-phase coordinate ascent framework as described in Algorithm 1. In the *Learning* phase, a dialog model is trained based on the available

[4]https://aws.amazon.com/elasticsearch-service/

assignments, and its weights are fixed (frozen). Then, in the *Assignment* phase, we compute a re-assignment that maximizes dialog model performance under different possible assignments. Searching over the complete document set is computationally infeasible, so we perform an approximate greedy search over possible documents ordered by their TF-IDF prior score.

We compare the performance of a Transformer [210] language model provided with grounding documents assigned by different algorithms in Figure 2.3a. A model without grounding scores by far the worst in terms of perplexity, which indicates that knowledge grounding is important for modeling media dialog. While TF-IDF assignments significantly improve performance compared to no grounding, probabilistic grounding models achieved the best performance. The sudden drops in perplexity values at every third epoch in Figure 2.3b indicates that the model was well-trained based on current assignments before a new assignments were obtained.

While our articles and conversations come from the same broadcasting source, the NPR interview transcripts generally do not contain links or metadata connecting them with specific grounding documents, and thus there are no ground truth labels available to us. To ascertain that the grounding is relevant, we enlisted two native English speakers who regularly listened to broadcast radio to perform a qualitative evaluation of 100 randomly sampled interview and article pairs. We found that 87% of these pairings are highly relevant, 5% are somewhat relevant and the rest are irrelevant. The inter-annotator agreement measured by Cohen's Kappa was 0.79. The lack of ground truth is something we would argue is not a limitation, rather our probabilistic linking step avoids the dependency on data that is not likely to be available in practice.

2.5 Modeling Media Dialog

A model's ability to learn underlying discourse dynamics is reflected in its performance on downstream tasks. Here, we assess how well our model learns from dialog structure and question-pattern metadata via utterance generation—a simple predictive task that relies on a

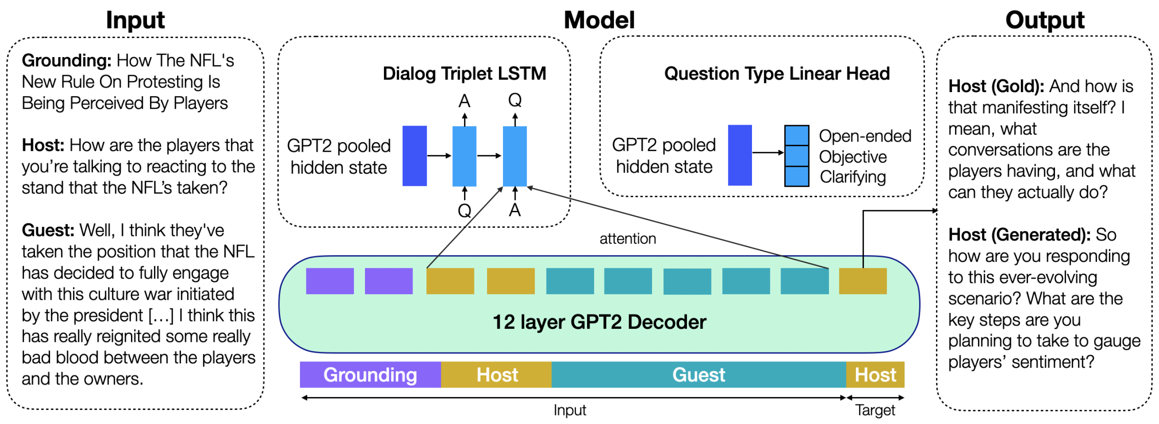

Input
Model
Output
Grounding: How The NFL's New Rule On Protesting Is Being Perceived By Players
Host: How are the players that you're talking to reacting to the stand that the NFL's taken?
Guest: Well, I think they've taken the position that the NFL has decided to fully engage with this culture war initiated by the president [...] I think this has really reignited some really bad blood between the players and the owners.
Dialog Triplet LSTM
GPT2 pooled hidden state
A
Q
Q
A
Question Type Linear Head
GPT2 pooled hidden state
Open-ended
Objective
Clarifying
attention
12 layer GPT2 Decoder
Grounding
Host
Guest
Host
Input
Target
Host (Gold): And how is that manifesting itself? I mean, what conversations are the players having, and what can they actually do?
Host (Generated): So how are you responding to this ever-evolving scenario? What are the key steps are you planning to take to gauge players' sentiment?

Figure 2.4. Knowledge grounded generator model with two discourse-specific auxiliary tasks for media dialog

holistic understanding of grounding knowledge and a dialog history. This serves as an initial measure of understanding of discourse patterns and grounding even if dialog produced varies.

We treat knowledge-grounded response generation in the media dialog setting as a language modeling task: given a dialog history H and a grounding knowledge document K, we seek to predict the next utterance x by maximizing the likelihood $p(x|H,K)$. The dialog history is composed of turns spoken by both the interviewer and interviewee where each utterance is provided with the role annotation. We only model interviewer (*host*) responses, which aim to moderate the conversation via questions and follow-ups. To understand the effect of dialog structure and question types in response modeling, we introduce two *auxiliary losses* to influence generation—a multi-task setup that has seen success in goal-oriented dialog generation [130].

2.5.1 Knowledge Grounded Generator

We use a common decoder-only model for knowledge-grounded dialog generation [60]: GPT2 [169], a pre-trained Transformer decoder. We concatenate grounding documents, dialog history, and the target response as model input. To distinguish each section, we add jointly-learned segment embeddings ({`Grounding`, `Host`, `Guest`}) to each input token. We demonstrate in Table 2.4 that such segment embeddings are essential for this kind of dialog modeling. We only consider target tokens for cross-entropy loss calculation via conditional likelihood $p(x|H,K)$.

Table 2.3. Performance on auxiliary tasks: Dialog Pattern prediction and Question Type prediction. Our auxiliary tasks are mutually helpful: using both losses improves both pattern accuracy and question type F1 score.

Model	Dialog Pattern Accuracy	Question Type F1 Score
KGG + Probabilistic Grounding	38.5	68.8
+ Dialog Pattern	86.3	76.2
+ Question Types	**87.9**	**90.5**

2.5.2 Predicting Look-ahead Dialog Patterns

Following Section 2.4.1, we use a generative model to explore the role of response type triplets in structuring media dialog (stemming from an interviewer utterance [73]). Following response type triplets defined in Section 2.4.1, we predict the pattern of the dialog triplet beginning with the generated host question as an auxiliary predictive task alongside host utterance generation. We treat this as a sequence transduction task, employing an LSTM [75] decoder with an initial hidden state computed by mean-pooling GPT2 final layer hidden states. Consider $\mathbf{s}_i$ the i-th hidden state from the GPT2 decoder for a length L sequence; now for each hidden state $\mathbf{l}_i$ in the LSTM decoder, we also calculate attention over the GPT2 hidden states, where $\{\mathbf{s}_i\}$ are the keys and values, and $\mathbf{l}_i$ is the query, resulting in an attended vector. We concatenate this attended vector with the LSTM hidden state $\mathbf{l}_i$ and then project it to predict the dialog triplet sequence, maximizing the log-likelihood.

2.5.3 Predicting Question types

We further explore the impact of question types (dialog acts) via another auxiliary task: multi-label classification for host utterance question types [145]. We surmise that accurately predicting question types will help infer question framing and wording, improving generation fidelity. Much like dialog pattern prediction, we use a pooled representation of GPT2 hidden states. We produce a score for each of three question attributes—polarity, combativeness, and subjectivity—via a linear projection and optimize via binary cross-entropy loss.

2.6 Experiments

In our experiments, we seek to answering the following: 1) Does knowledge grounding help generate more topical host responses? 2) Do our two auxiliary discourse losses improve dialog generation performance? 3) Do human raters find responses generated by our model coherent and fluent?

Network Architecture

For probabilistic linking, we use a 6-layer encoder-decoder Transformer model [210]. The input to the model consists of grounding document followed by dialog history. The output is the next response in the dialog. To speed up the learning phase, we use ReZero initialization [6] that do not require learning weight warm-up schedule. We also observe that performing reassigning at every epoch results in noisy update in assignments and weaker local optima is achieved at the end. When we switch the reassignment phase for every third epoch, the learning stabilizes mirroring a line search [225] from coordinate descent optimization. For the media dialog generation model, we use GPT2 [169] (Transformer with 12 layers, 768 hidden size, 12 heads, and 117M parameters— `gpt2-small`) as the base architecture. Our best model, KGG with two discourse-specific auxiliary losses, has 124M parameters.

Hyperparameters

We use history size 5 and number of grounding documents as 5. We use the RAdam optimizer [124] and the learning rate was set at $6.25e - 5$ with a linear decay of step size 10^{-1} per epoch. The loss coefficients in the multi-task loss function for dialog modeling loss, dialog pattern prediction loss and question type prediction loss were 2.0, 1.0, and 1.0 respectively. Each model converged in 3 epochs on average with batch size 4 in a TITAN X (Pascal) GPU that took 6 hours in total. While training, we only observe perplexity on the validation set to employ an early-stopping criteria.

Table 2.4. Metrics on generated interviewer responses on test set. NPO/NEO = Noun-phrase/Named entity overlap with context (C) and gold (G); QR = Question rate. NIDF is a measure of specificity [184]. QR, NPO, NEO are measured in percentages.

Model	PPL	BLEU	QR	NPOG	NPOC	NEOG	NEOC	NIDF
No Grounding								
Finetuned (FT) GPT2	28.6	15.4	34.2	0.67	0.57	0.92	0.98	0.105
FT GPT2 + Segment	27.5	17.5	49.9	1.70	1.67	1.56	1.55	0.117
Effect of grounding								
MemNet [39] + TF-IDF	26.5	17.8	43.8	1.86	1.63	1.51	1.62	0.187
MemNet [39] + Prob. Ground.	25.1	17.7	46.9	1.98	2.31	2.89	3.02	0.197
KGG (TF-IDF)	23.5	18.1	48.5	2.73	3.91	3.01	5.58	0.245
KGG (Prob. Ground.)	19.6	19.2	53.6	3.24	4.67	3.44	6.78	0.267
Auxiliary Losses								
+ Dialog Pattern	17.2	**21.0**	56.7	3.52	**6.92**	5.16	**7.85**	0.302
+ Question Types	**15.8**	20.3	**58.9**	**3.67**	6.79	**5.89**	7.79	**0.359**

Metrics

To measure the fidelity of generated responses, we compute BPE perplexity and BLEU [156] between generated and gold utterances. To assess topical accuracy, we calculate the overlap between noun-phrases and named entities in the generated and gold responses. We are also interested in measuring coherence with respect to the context (i.e., grounding documents and dialog history), calculated via the noun-phrase and named entity overlap between generated responses and context. Furthermore, as news interviews are intended to inform audiences, interviewers must ask questions using specific vocabulary and construction. To assess this, we adopt the Normalized Inverse Document Frequency [184] to measure vocabulary specificity via word rarity. As we focus on generating interrogative host responses, we also calculate the percentage of questions asked in the generated responses as a measure of model inquisitiveness.

2.6.1 Effect of Knowledge Grounding

To assess the usefulness of explicit grounding documents, we first compare dialog models that use and do not use such documents in Table 2.4. Using segment embeddings to mark utterance bounds improves all measures of fidelity, signifying that this is a useful way to leverage

Table 2.5. Sample generated response on NFL's new rule. When we add discourse specific losses, the models generate questions that bears more coherence to the context as well as ask clarifying questions.

Grounding (PL)	How The NFL's New Rule On Protesting Is Being Perceived By Players
Context	**HOST**: How are the players that you're talking to reacting to the stand that the NFL's taken? **GUEST**: Well, I think they've taken the position that the NFL has decided to fully engage with this culture war initiated by the president [. . .] I think this has really reignited some really bad blood between the players and the owners. **HOST**:
Gold	And how is that manifesting itself? I mean, what conversations are the players having, and what can they actually do?
KGG (TF-IDF)	Can you tell me more about NFL's new rules?
KGG (Probabilistic)	This are some significant changes in NFL's rules. I think the most effect will be on the players.
+ Dialog Pattern	Okay so let's talk about NFL players. What is your gameplan?
+ Question Types	So how are you responding to this ever-evolving scenario? What are the key steps are you planning to take to gauge players' sentiment?

speaker role information in dialog modeling. Models that use external grounding knowledge outperform non-grounded models by 1-8 points on almost all metrics, suggesting that such grounding is an important component of host response generation models. To assess the impact of our knowledge grounded generator (KGG) architecture, we compare performance against a strong Memory Network (MemNet) baseline for knowledge grounded dialog generation [39]. We confirm our choice of a GPT2-based KGG, as it outperforms Memory Networks in all metrics.

Next, we compare the impact of document assignments made via TF-IDF and our probabilistic linking (PL) method. We once again see improved fidelity, mirroring our observations from Section 2.4.3. Models trained using PL document assignments generate utterances with 19-20% higher noun-phrase and named entity overlap with the gold utterance and context, indicating that PL assignments allow the KGG to more strongly condition on the provided context.

2.6.2 Effect of Auxiliary Tasks

In this experiment, we investigate how predicting dialog patterns and question types impacts the specificity and fidelity of generated host responses. Each auxiliary loss contributes a significant improvement (1-2 points) in perplexity but affects fidelity and topicality in different ways. With dialog pattern prediction, we observe that generated responses are more coherent with respect to conversational context, seeing 8% and 48% improvements in noun phrase and named entity overlap with dialog history, respectively. This supports the sociolinguistic observation that the interviewer's choice of utterance (i.e., whether to ask a question, and response content) depends on the discourse structure toward which they aim to guide the conversation [73]. Our results suggest that biasing a dialog model to predict future discourse structure can encourage it to more effectively leverage the past dialog structure (from the conversation history). We confirm in Table 2.3 that this model can predict look-ahead dialog patterns with 86.3% test-set accuracy. In light of findings that vanilla dialog models may not condition well on conversation context [181], our results suggest one possible direction toward improving contextual language modeling for dialog with inherent structure, such as media dialog.

When we add question-type-prediction loss, we see a significant drop in perplexity and improved fidelity. As expected, by inducing our model to predict the question attributes for the target utterance, our model achieves the highest inquisitiveness (58% question rate). It can also accurately predict question types, with 90.5% macro-averaged test set F1 score. Our results suggest that as the model learns to categorize the interviewer response via specific attributes, it simultaneously learns to generate responses with more specific wording. Table 2.5 and Table 2.6 contain representative generations from our best model as well as other baselines, showing that when we add additional discourse specific losses, our model appropriately captures the interviewer's clarifying intent and conversation direction.

Table 2.6. Sample generated response on nuclear threat. KGG with discourse specific losses generate more specific and on-topic responses.

Grounding (PL)	Trump's Week Of 'Fire And Fury'
Context	**GUEST**: Steve Bannon is quoted as saying [. . .] the president has lost it. Now, are you supporting a president who is incapable of being entrusted with [. . .] nuclear weapons? **GUEST**: Well - one thing I haven't heard yet is Steve Bannon interviewed [. . .] so look, I think the president of the United States has shown he's very, very capable [. . .]. **HOST**:
Gold	Should he be taunting a dictator with nuclear weapons about the size of his nuclear button?
KGG (TF-IDF)	Well, that's what you've been talking about, right?
KGG (Probabilistic)	What do you see as a future?
+ Dialog Pattern	I am worried about his political position now.
+ Question Types	Do you think it's a good idea to confront a nuclear war?

2.6.3 Human Evaluation

Automatic evaluation of dialog generation quality is still unreliable [122, 154], and thus we provide evaluation by human users. We perform pairwise comparisons between responses generated by our best system and those generated by four strong baselines: the best model with no grounding, KGG with TF-IDF, KGG with PL, and KGG with dialog pattern prediction. We also compare against the gold response. Our human evaluation study (details in Appendix §B) measures three aspects of response quality on 100 test examples: 1) How relevant the response is with respect to **dialog history**; 2) How relevant the response is with respect to **grounding documents**; and 3) Whether the generated response is **fluent** English.

We observe in Table 2.7 that human judges prefer responses generated by our best model (with both discourse analysis auxiliary tasks) to baselines by statistically significant margins in almost every case. This indicates that dialog structure and question types are highly useful for generative modeling in a media dialog setting—specifically news interviews. Human raters also found that despite a significant drop in perplexity when adding the question-type prediction loss,

Table 2.7. Pairwise comparison between responses generated by our best model (with both discourse analysis auxiliary tasks) vs. responses generated by other baselines as well as the Gold response. All numbers are in percentages (highest underlined). Ties are not shown. **Bold** entries denote significance with $p < 0.05$ from bootstrap tests on 1000 subsets of size 50.

Best Model vs. →	**No Ground**		**TF-IDF**		**Prob. Ground.**		**+ Dialog Pat.**		**Gold**	
Metric ↓	W	L	W	L	W	L	W	L	W	L
Relevance (Dialog)	**85.1**	9.2	**86.5**	3.3	**69.1**	27.6	**61.0**	22.4	36.7	**47.4**
Relevance (Grounding)	**94.5**	4.3	**91.6**	1.9	**82.5**	4.7	84.5	9.5	30.1	**55.2**
Fluency	**97.2**	0.8	**87.1**	7.8	62.1	10.1	58.7	11.2	20.8	**24.6**

the two versions of discourse-conditioned models had similar fluency, indicating similar language modeling performance. We observe an inter-annotator agreement (Cohen's kappa) of 0.79, 0.92, and 0.73 for relevance to dialog history, grounding documents, and fluency, respectively.

2.7 Conclusion

In this chapter, we perform the first large-scale analysis of discourse patterns in media dialog, using a new dataset of 23K annotated news interview transcripts: INTERVIEW. Our results mirror findings from linguistic studies of news interviews [221, 73]. We demonstrate that adding auxiliary tasks for discourse pattern and interrogative type prediction helps model such media dialog. We observe that responses depend heavily on external knowledge, and present a probabilistic framework for linking factual documents with a conversation. While we focus on discourse *pattern* analysis, INTERVIEW also supports analysis of temporal patterns in interviewing, argumentation, and knowledge grounding in long conversations.

2.8 Acknowledgements

This chapter contains material as it appears in and adapted from Empirical Methods in Natural Language Processing, 2020 ("Interview: Large-scale Modeling of Media Dialog with Discourse Patterns and Knowledge Grounding" Bodhisattwa Prasad Majumder*, Shuyang Li*, Jianmo Ni, Julian McAuley). The dissertation author was co-primary investigator and author.

Chapter 3

Type-Aware Question-Answering for Zero-shot Dialog State Tracking

3.1 Introduction

Dialog agents are gaining increasing prominence in daily life. These systems aim to assist users via natural language conversations, taking the form of digital assistants who help accomplish everyday tasks by interfacing with connected devices and services. A key component to understanding and enabling these task-oriented dialogs is Dialog State Tracking (DST): extracting user intent and goals from conversations via filling in belief slots [106, 219] (Figure 3.1). Assistive and recommendation use-cases for dialog agents in production settings are particularly challenging due to constantly changing services and applications with which they interface. **In this chapter,** we aim to develop a natural-language, type-centric framework for inferring user preferences and intents in new conversations and domains, as well as a pre-training methodology to instill type awareness and knowledge in language models.

Traditional DST systems have achieved high accuracy when presented with a known ontology of slot types and valid values [25]. In a real-world setting, however, a DST model must generalize to new slot *values* (e.g. new entities that are not present at training time) and new slot *types* (e.g. requirements regarding a new application). Recent work has sought to address these issues by posing DST as a reading comprehension or question answering (QA) task [54]—such models predict each slot value independently at any given turn and can theoretically be queried

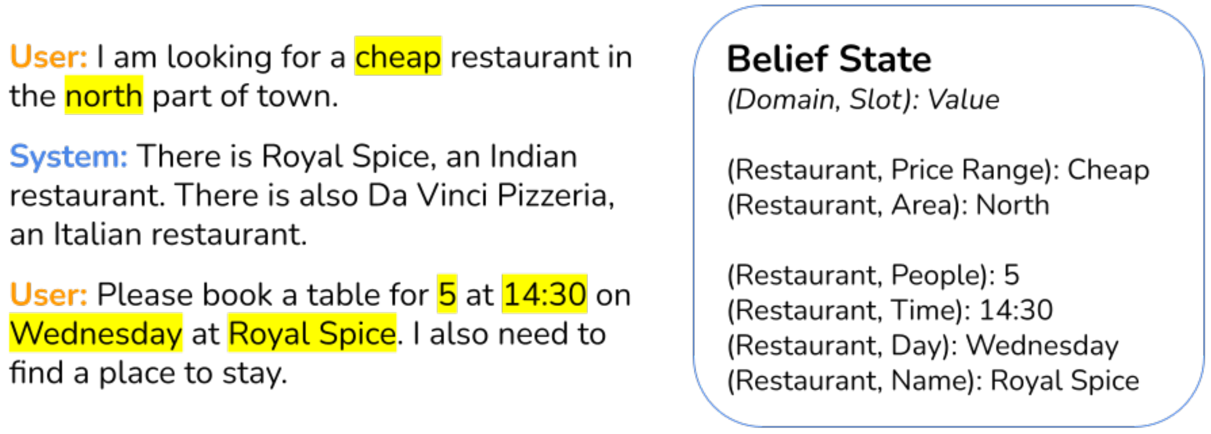

User: I am looking for a cheap restaurant in the north part of town.
System: There is Royal Spice, an Indian restaurant. There is also Da Vinci Pizzeria, an Italian restaurant.
User: Please book a table for 5 at 14:30 on Wednesday at Royal Spice. I also need to find a place to stay.
Belief State
(Domain, Slot): Value
(Restaurant, Price Range): Cheap
(Restaurant, Area): North
(Restaurant, People): 5
(Restaurant, Time): 14:30
(Restaurant, Day): Wednesday
(Restaurant, Name): Royal Spice

Figure 3.1. In Dialog State Tracking (DST), we aim to infer a user's preferences about certain *slots* in a conversation. Preferences expressed by the user are highlighted in the conversation transcript on the left, while the corresponding cumulative belief state is on the right. In the zero-shot setting, we aim to infer user preferences from conversations in *new*, unseen domains.

for new slots at inference time.

Some approaches toward DST as QA learn embedding vectors for each slot and/or domain word [226], but this is not robust to unseen slots whose specific names (e.g. 'Internet Access') may be totally unlike those in the training set. Gao et al. [55] attempt to remedy this by posing a natural language question for each slot, but their hybrid span-extraction and classification-based system nonetheless requires access to the full ontology for unknown domains. We present an ontology-free model using natural language questions to represent slots that builds on conditional language modeling techniques—taking advantage of the rise of powerful generative language models [168]—to tackle DST as a *generative* QA task. Our model can generalize to unseen domains, slot types, and values, and allows developers to query for arbitrary user requirements via simple questions.

Our question-answering framework helps reveal the parallels between dialog state tracking and entity typing tasks—identifying user's preferences about *slots* bears close similarity to identifying specific entities of a given *type* or category mentioned in a text (e.g. that the user expressed interest in an "Indian" restaurant, where "Indian" is a type of "cuisine"). To better understand user opinions in unseen domains, we leverage the fact that entities can be categorized

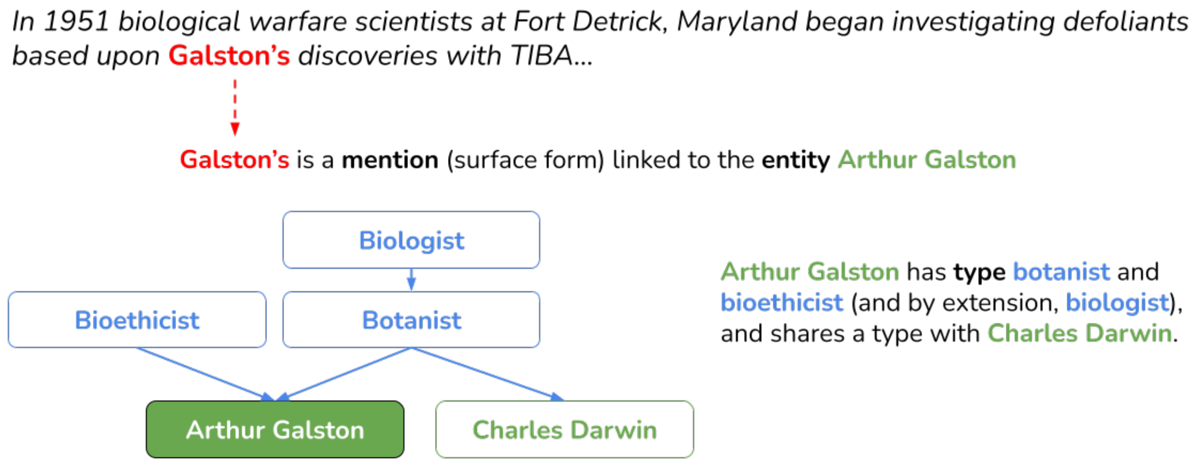

In 1951 biological warfare scientists at Fort Detrick, Maryland began investigating defoliants based upon Galston's discoveries with TIBA...
Galston's is a mention (surface form) linked to the entity Arthur Galston
Biologist
Bioethicist
Botanist
Arthur Galston
Charles Darwin
Arthur Galston has type botanist and bioethicist (and by extension, biologist), and shares a type with Charles Darwin.

Figure 3.2. Via the **WikiWiki** dataset, we train a model to answer questions about entities mentioned in Wikipedia articles (top) and WIkidata *types* that such entities are an *instance of* (P31) or *subclass of* (P279).

by their *types*—which indicate where they belong in a taxonomy of knowledge. Much like how knowledge acquisition in cognitive development progresses from recognizing concrete objects to gradually understanding their relations to one another [131], we aim to extend language models' existing rough understanding of entities [71] to the types that govern how entities are related. Instilling type knowledge in multi-purpose models can improve performance in tasks like entity linking [155], question-answering [49], and semantic parsing [201]. We demonstrate in this chapter that type-aware language models also confer significant improvements in domain generalization for DST.

While existing language models can memorize some facts [161], they frequently hallucinate false information [127, 191]. Current attempts to learn to infer types for entities are hampered by 1) the difficulty of collecting diverse, large-scale typing datasets; and 2) how existing corpora assume independence between types [29], while in reality types sit at levels of granularity that are useful in different settings: a pizza store may care whether a user likes Cheese Pizza; a restaurant recommender might care if the user wants Pizza; finally, a general dialog agent might only care if a user wants Food.

We address both issues by proposing a simple and effective approach for pre-training generative language models to answer questions about entities, types, and surface forms (mentions) in a large public knowledge graph (KG) consisting of Wikipedia articles and Wikidata nodes. We leverage high quality type labels in a large corpus of knowledge-rich text *and* an ordered, hierarchical type ontology.

To summarize our main contributions in this chapter:

- We propose an ontology-free conditional language modeling framework for dialog state tracking via generative question answering, achieving state-of-the-art performance in zero-shot domain adaptation settings for DST on MultiWOZ 2.1 [42] across all domains with average per-domain gains of 5.9% joint accuracy over previous best methods;
- We show that our approach can be easily adapted to predict slot carry-over and transfer knowledge from a larger, more diverse dataset [93], improving zero-shot DST performance across all domains to 11% joint accuracy over the state-of-the-art;
- We create the new **WikiWiki** dataset comprising 10M Wikipedia articles linked to nodes from Wikidata and propose a pre-training scheme for generative language models using type-centric question-answering based on WikiWiki;
- We show that our models can precisely infer types for seen and unseen entities in new articles from WikiWiki, and propose novel types that humans judge to be accurate and appropriate; and
- We achieve state-of-the-art (SOTA) performance in zero-shot domain adaptation for dialog state tracking using our type-instilled models, with average per-domain gains of 20.8% joint accuracy.

3.2 Related Work

Dialog State Tracking

Modern dialog state tracking seeks to capture evolving user intents in a structured belief state [202]. Traditional systems rely on hand-crafted features [72] and classify slot values from a fixed ontology [149, 172]. Gao et al. [54] and Zhou and Small [250] fill some slots via spans extracted from dialog history, although they treat non-numeric slots as categorical. Generative methods [230, 226] can predict arbitrary unseen values, with Hosseini-Asl et al. [77] achieving state-of-the-art supervised DST performance in MultiWOZ 2.1 although they cannot predict unseen slots. By posing DST as generative QA, our framework can leverage language models pre-trained on open-domain documents [168] to understand unfamiliar queries. Like Gao et al. [55], we seek to answer natural language questions about each slot. We contrast our approach to zero-shot DST—which never has access to slots or dialog from the target domain—and that of Campagna et al. [20], who expose their 'zero-shot' models to synthetic in-domain conversations that require access to the full ontology of the 'held-out' evaluation domain.

Question Answering

We take inspiration from previous work that frames a wide selection of natural language understanding (NLU) tasks [215] as QA [144] and span extraction [90]. While question-answering can be posed as a span extraction task [216], generative approaches have proven successful in answering questions about complex passages [48]. We use a language modeling approach, taking cues from Raffel et al. [170] who demonstrate that a large language model trained on next-token prediction can learn to solve many different NLU tasks posed as text. Recent work has also shown that large pre-trained language models can generalize to new NLU tasks with few or no examples [16], and we leverage this alongside world knowledge acquired during the pre-training process [161] to build a DST model that is robust to new domains and slot-value ontologies.

Knowledge Grounding in Language Models

Large pre-trained language models have been shown to memorize some facts [161]. One recent line of work aims to explicitly condition generation on knowledge bases by combining a retrieval module and a language model [138, 66, 109, 141]. Peters et al. [160] propose instead to align token representations from pre-trained language models with entity embeddings to reason over a limited set of entities. Yamada et al. [232] explicitly denote entity tokens with a learned input embedding. Specific entity embeddings have also been learned jointly by using knowledge graphs as auxiliary inputs during language model pre-training [197, 49, 244]. Another line of work aims to model specific factual statements from knowledge bases [218] for reading comprehension [129] and trivia QA [1]. We propose text-to-text pre-training on knowledge recovery tasks to instill *type-awareness*. Our models learn type knowledge that transfers to the type-adjacent downstream task of dialog state tracking and can infer unseen types.

Entity Representation Learning

Many SOTA systems for knowledge retrieval and QA rely on learned dense embeddings of individual entities or types to perform multi-class classification [53, 85, 228]. Several recent frameworks aim to learn entity knowledge during language model pre-training via entity masking [198] or contrastive learning [163]. Systems for entity typing [36] and disambiguation [231] also learn dense vector encodings that are later matched via dot-product scoring. Cao et al. [21] aim to address some downsides of the above approaches—the linearly increasing space required to store learned representations and difficulties in negative sampling—by casting the task as generative language modeling: predict the name of an entity to be linked. We generalize this approach from entity names (which appear verbatim) to include types, which require a more nuanced understanding of a context.

3.3 A Generative Question Answering Approach for DST

We follow Gao et al. [54] in treating Dialog State Tracking as a reading comprehension problem: at each turn of dialog, our model reads the dialog history and answers a fixed set of queries about user requirements and preferences (slots), with predictions aggregated to form the belief state. In our framework (Figure 3.3), we query for a given slot (e.g. Hotel Price Range) by asking a natural language question [55]—"What is the price range of the hotel the user prefers?". As our model's predictive ability is based on its general understanding of language and task-oriented conversation, we support zero-shot inference without the need to re-train the

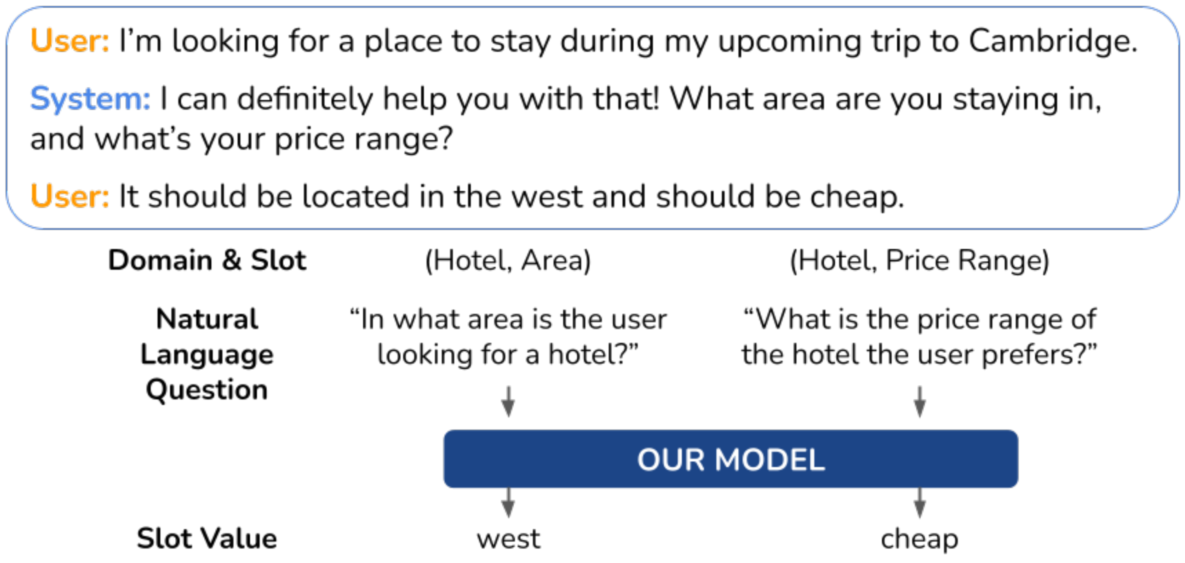

User: I'm looking for a place to stay during my upcoming trip to Cambridge.
System: I can definitely help you with that! What area are you staying in, and what's your price range?
User: It should be located in the west and should be cheap.
Domain & Slot
(Hotel, Area)
(Hotel, Price Range)
Natural Language Question
"In what area is the user looking for a hotel?"
"What is the price range of the hotel the user prefers?"
OUR MODEL
Slot Value
west
cheap

Figure 3.3. Based on a dialog history, a natural language questions are provided to our model to query a user's requirements and preferences (dialog state). We construct questions to resemble colloquial human speech, in contrast to approaches that use templated questions [120].

model or extend a formal ontology. For example, if a model has not been trained on data from the hotel domain, when presented with a hotel booking conversation we may nonetheless ask it a question like "In what area is the user looking for a hotel?" and received a prediction for that unseen requirement (Hotel Area).

While we conduct our experiments on English-language DST datasets, our approach is applicable to state tracking in any language, provided a conversation history is available.

3.3.1 Problem Statement

We consider a conversation with T turns of user u_t and system utterances y_t: $C = \{y_1, u_1, \ldots y_T, u_T\}$. The belief state B_t at turn t comprises many tuples of slots $s \in S$ and their associated values $v_{s,t} \in V_s$, extracted from the conversation history $C_t = \{y_1, u_1, \ldots, y_t, u_t\}$. The set of possible values V_s can be arbitrarily large (e.g. possible hotel names), so we represent these values as sequences of vocabulary tokens $v_{s,t} = \{w_1, w_2, \ldots, w_k\}, w_i \in \mathcal{W}$. At inference time we pose a natural language question $s = \{w_1, \ldots, w_n\}$ and our model predicts an answer (slot value $v_{s,t}$) based on its understanding of the dialog history C_t. To predict the belief state B_t, our model independently answers $|S|$ different questions (Figure 3.3). In zero-shot DST, the system must predict values for slots outside of the initial ontology—these slot queries correspond to arbitrary natural language questions s' about entities and relationships in the conversation C_t.

3.3.2 Generalizing to New Domains and Slots

Dialog State Tracking systems in real-world settings must scale to new users and services, accommodating new slot values (e.g. a new movie release) as well as new domains and slot types (e.g. a service update, or a new connected API). Existing methods require the developer to either write a complete ontology of slots and allowed values or modify their model architecture to add slot-specific prediction heads [25]. Span-based approaches [241, 250] can correctly predict values that appear verbatim in a conversation but fail when a user paraphrases or mis-phrases a value. They also fall back to treating open-valued slots as classification problems [241, 55]. We approach DST as an ontology-free generative question answering task, as generative methods [226, 99] have shown promise in few-shot and supervised DST settings.

While some approaches toward DST as QA learn a set of embeddings for each slot and/or domain [54, 226, 99], this is not robust to unseen slots. We encode slots as natural language questions—manually formulating one question per slot—allowing us to share a pre-trained encoder for both dialog context and slot to leverage shared linguistic knowledge [55]. Thus, our model is also agnostic to ontologies and can answer arbitrary English questions about the dialog history. We treat DST via QA as a conditional language modeling task, and train our model to predict the conditional likelihood of question (slot s) and answer (value $v_{s,t}$) tokens given a dialog context C_t at a given turn t:

$$P(v_{s,t}, s|C_t) = P(v_{s,t}|s, C_t) * P(s|C_t)$$

At inference time, the model is given the dialog context alongside a question—$[C_t; s]$—and asked to predict the value $v_{s,t}$ for that slot.

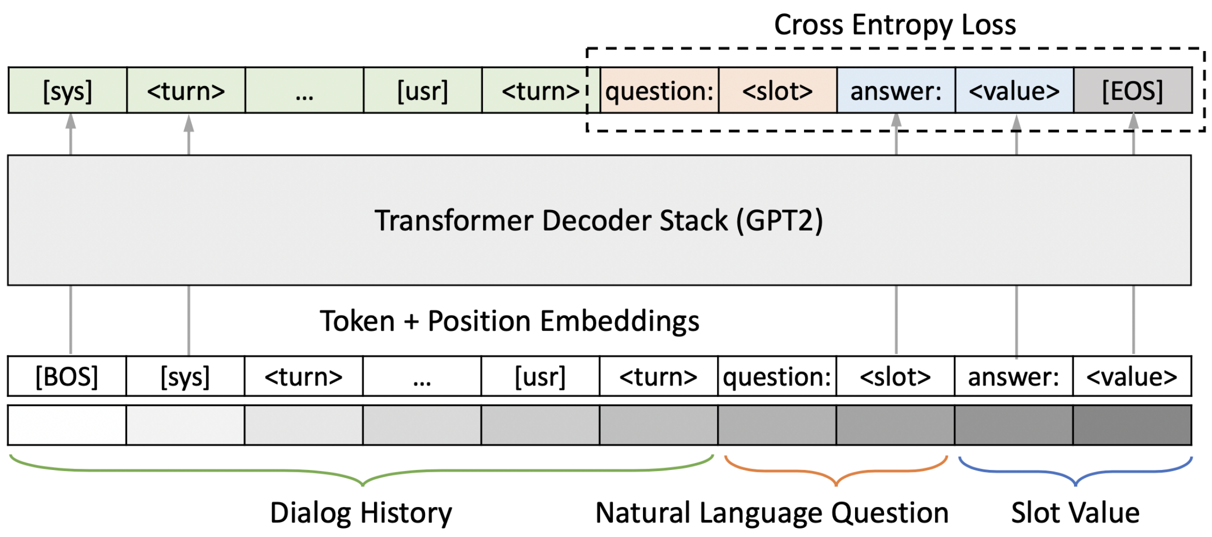


Figure 3.4. Model architecture of GPT2-DST. We model a single sequence with: the dialog history, a natural language slot query, and the slot value. Natural language questions for dialog slots allow our model to generalize to new slots via its understanding of general language.

3.4 GPT2-DST Model

For our conditional language model, we use an auto-regressive Transformer [210] decoder language model. This allows us to leverage pre-trained language models like GPT2 [168] and common-sense world knowledge accrued through pre-training [161].

We use a BPE [185] tokenizer to convert input text into a sequence of tokens. These are embedded in $\mathbb{R}^h$ and added to an $\mathbb{R}^h$ sinusoidal positional embedding. This input embedding is processed by l Transformer layers with hidden dimensionality h, each of which applies multi-headed attention with k heads followed by a feed-forward layer with a softmax nonlinearity. The final output hidden states are then projected into our vocabulary space of 50,257 sub-word tokens. We initialize our model weights with DistilGPT2 [180], GPT2 [168], or GPT2-medium with $h = 768, 768, 1024$, $l = 6, 12, 24$, and $k = 12, 12, 16$ respectively.

As seen in Figure 3.4, our input sequence consists of a concatenation of dialog context C_t, slot query s, and slot value $v_{s,t}$: $[C_t; s; v_{s,t}]$. We pre-pend each utterance with a speaker token `[usr]` or `[sys]` for a user or system speaker to allow our model to identify additional context about each utterance. We pre-pend the slot query and value with `question:` and `answer:`

respectively to distinguish slot queries from user-posed questions in the conversation. At training time, we calculate a cross-entropy loss similar to encoder-decoder models by maximizing the log likelihood of the slot query and value conditioned on the dialog context:

$$P(s, v_{s,t}|C_t) = \prod_i^n P(x_i|x_{<i}, C_t)$$

where $n = |[s; v_{s,t}]|$. We find through ablation experiments on our architecture that this loss computation method out-performs a naïve language-modeling approach that maximizes log likelihood of the full concatenated sequence $[C_t; s; v_{s,t}]$ via the factorized joint distribution [159, 77]:

$$P(x) = \prod_i^n P(x_i|x_{<i})$$

This allows for flexibility in learned representations for dialog context while regularizing slot query hidden states.

3.5 Dialog Data

We perform our experiments on **MultiWOZ** [17], which contains over 10K single- and multi-domain task-oriented dialogs written by crowd-workers. We use the 2.1 version, with corrected and standardized annotations from Eric et al. [42]. We follow Wu et al. [226] in lower-casing all dialogs and removing dialogs from training-only domains (Police and Hospital). The final dataset contains 9,906 conversations from 5 domains (Restaurant, Hotel, Attraction, Train, Taxi) covering 30 domain-slot pairs. Each dialog contains on average 7 user / system turns.

We also experiment with augmenting our training dataset in zero-shot settings with observations drawn from the DSTC8 [93] dataset,[1] which contains 16,152 dialogs from 45 domains. DSTC8 was created via template-based dialog models provided with service APIs, and

[1]https://github.com/google-research-datasets/dstc8-schema-guided-dialogue

Table 3.1. Dataset statistics for MultiWOZ 2.1 and DSTC8: number of dialogs in each split, number of domains, and slots with slot category breakdowns.

					Slots				
	Train	Dev	Test	Domains	Open	Numeric	Temporal	Categorical	Total
MultiWOZ	7.9 K	1 K	1 K	5	9	5	5	11	30
DSTC8	16.1 K	2.5 K	4.2 K	19	59	12	10	43	124

then edited by crowd-workers [186]. We normalize domains and slots corresponding to the same domain (e.g. `Bus_1, Bus_2`) for a total of 19 domains and 124 slot types in DSTC8. We further manually annotate each dataset with slot value types: open-valued (e.g. Hotel Name), numeric (e.g. Restaurant Guests), temporal (e.g. Taxi LeaveAt), and categorical (e.g. Attraction Type). Dataset statistics are shown in Table 3.1.

3.6 Experiments

We measure DST performance via Joint Goal Accuracy (JGA): the proportion of turns with all belief slots predicted correctly, including those not present. In Section 3.6.1, we evaluate our model on fully *supervised* DST, in which all domains and slots are known at training time. In Section 3.6.2, we investigate *zero-shot* domain adaptation in which the model is evaluated on conversations from an unseen domain with previously unseen slots. We then explore how our framework seamlessly accommodates teaching a model to predict slot carry-over (Section 3.6.4) and transfer learning with significantly more diverse domains and slot types (Section 3.6.5). To measure zero-shot JGA, we follow Campagna et al. [20] and only consider slots specific to the held-out domain. We focus our analysis on the zero-shot setting, as our goal is to build DST systems that can easily and effectively generalize to new domains and services. We train all models to convergence with a maximum of 10 epochs on Nvidia V100 GPUs, using the Lamb optimizer [240] with a base learning rate of 2e-5. All predictions are made using greedy decoding.

Table 3.2. Supervised DST performance on MultiWOZ 2.1 of our model (underlined) compared to (a) prior methods capable of zero-shot inference (**SUMBT and STARC require access to slot-value ontologies at inference time*), with models using natural language questions (NLQ) annotated; and (b) SOTA models for supervised DST that are incapable of zero-shot inference.

(a) Methods capable of zero-shot DST

Model	Type	JGA	NLQ
TRADE [226]	G	45.60	
SUMBT [103]*	C	46.70	
STARC [55]*	C+S	49.48	Y
MA-DST [99]	G	51.88	
<u>GPT2-DST</u>	G	**52.58**	Y

(b) SOTA methods for supervised DST

Model	Type	JGA	Extra Supervision
DSTQA	C+S	51.17	Knowledge Graph
DS-DST	C+S	51.21	
<u>GPT2-DST</u>	G	52.58	
SOM-DST	G	53.68	Previous Dialog State
SST	C	55.23	Schema
TripPy	S	55.30	Previous Dialog Actions
SimpleToD	G	**55.72**	Actions (Training)

Table 3.3. Ablation study of our framework, reporting supervised JGA on MultiWOZ 2.1.

Base Model	Approach	Supervised JGA (%)	# Params
DistilGPT	Language Modeling	36.35	82 M
	Conditional Language Modeling (no Pre-training)	39.34	
	Conditional Language Modeling	49.55	
	Conditional Language Modeling + Question	<u>50.83</u>	
GPT2	Conditional Language Modeling + Question	51.02	124 M
GPT2-medium	Conditional Language Modeling + Question	**52.58**	355 M

3.6.1 Supervised DST

We first evaluate on the commonly benchmarked supervised DST task to demonstrate performance competitive with state-of-the-art. In this setting we compare our approach against prior methods capable of zero-shot inference in Table 3.2a—TRADE, STARC, SUMBT, and MA-DST—and those incapable of doing so in Table 3.2b, including DSTQA [250], DS-DST [241], SOM-DST [94], SST [25], TripPy [70], and SimpleToD [77]. Our model outperforms all prior models that support zero-shot generalization and is competitive with methods that focus solely on supervised DST—most of which require extra supervision at training and inference time, including dialog actions and prior dialog states. We distinguish models by their prediction type as (C)lassification-, (S)pan extraction-, and (G)eneration-based methods.

As seen in Table 3.3, our formulation of DST as a generative QA task benefits significantly from the usage of a conditional decoder-style model. A standard auto-regressive language modeling formulation (**Language Modeling**) with loss computed over the entire input sequence achieves 13% lower JGA compared to computing cross entropy loss only over slot value tokens (**Conditional Language Modeling**). Pre-training is also crucial—we see a 10-point drop in JGA when randomly initializing model weights (**no Pre-training**) compared to initializing from pre-trained DistilGPT2 weights. We also compare two other sizes of our models: GPT2-based—comparable in size to TRADE's [226] ~100M parameters— and GPT2-medium-based—comparable in size to STARC's [55] 355M parameters. We find that scaling the size of our model results in modest improvements in supervised JGA. We hypothesize that extending our loss to cover both slot query and value tokens (**+Question**) helps regularize the hidden representations of question tokens, and we achieve a 1.3% improvement in JGA.

3.6.2 Zero-Shot DST

Our primary focus lies in the zero-shot domain adaptation setting, where conversations and target slots at inference time come from unseen domains. We use a leave-one-out setup, training our models on four domains from MultiWOZ and evaluating on the held-out domain. Our model must understand a wide variety of possible questions about unseen conversations to generalize well. We compare our model against strong baseline models for zero-shot DST: TRADE, SUMBT, and MA-DST; Table 3.4 contains results from our models alongside baseline results reported by Kumar et al. [99] and Campagna et al. [20]. These models represent slots as domain-slot tuples: TRADE learns a separate embedding for each domain and word in slot names, while SUMBT and MA-DST encode domain-slot tuples via BERT [38] and an RNN encoder, respectively.

Our GPT2-medium based model achieves state-of-the-art zero-shot performance on all five domains, and by a significant (5-10%) margin on the Restaurant, Hotel, Attraction, and Train domains. While increased model size modestly impacts supervised DST performance (Table 3.3),

Table 3.4. Zero-shot domain adaptation JGA (%) on MultiWOZ 2.1 test set for recent works and our models with question loss, on the Restaurant, Hotel, Attraction, Train, and Taxi domains. Previous state-of-the-art results are underlined, with new best **bolded**.

	Restaurant	Hotel	Attraction	Train	Taxi
TRADE	12.59	14.20	20.06	22.39	59.21
MA-DST	13.56	16.28	22.46	<u>22.76</u>	59.27
SUMBT	<u>16.50</u>	<u>19.80</u>	<u>22.60</u>	22.50	<u>59.50</u>
GPT2-DST (small)	21.05	18.54	23.67	24.34	59.10
GPT2-DST	**26.17**	**24.41**	**31.31**	**29.07**	**59.61**

larger models perform significantly better in a zero-shot setting with average absolute gains of 4.8% and relative gains of 22% in JGA across domains. Such improvements are consistent with findings from Brown et al. [16] that up-sizing language models improves zero-shot performance across various tasks and Petroni et al. [161], who observe that larger pre-trained models can retain more common-sense and world knowledge from their pre-training corpus—which may help our model understand queries for unseen domains and slots.

Effect of Natural Language Questions

Prior work that frames DST as QA typically represents the slot query as a concatenation (tuple) of domain and slot name. Zhang et al. [241] explore the impact of three different slot representations—domain-slot tuples, short slot descriptions, and full questions—on a hybrid classification-extraction model for DST, and find little difference in performance. However, we find that full questions work much better than domain-slot tuples for our generative framework, especially in zero-shot DST. We hypothesize that natural language questions—structurally similar to dialog utterances and pre-training sentences—allow our model to best leverage its linguistic knowledge with minimal friction when jointly encoding the dialog history, slot query, and value.

Wu et al. [226] find that zero-shot generalization in models that represent slots as tuples is primarily due to shared slot names between domains (e.g. Taxi and Train 'leaveAt'). In a real-world setting a newly added dialog service is unlikely to share slot names verbatim with existing services. To fairly compare tuples and natural language questions under our framework,

Table 3.5. Example of different classes of DST errors, and the proportion of errors they make up across the four slot categories—Open, (Num)erical, (Temp)oral, and (Cat)egorical—for all five domains in a zero-shot setting. Latest (target) turn is **bolded**.

USER	My friend told me about Carolina Bed and Breakfast. Do you know about it?						
SYS	It's a 4 star guesthouse. What would you like to know about it?						
USER	Can you give me the postcode? And, do they have internet?						
SYS	The postcode is cb13nx; they have internet.						
USER	Thanks. Any boat attractions in the west?						
SYS	**Nothing in west. Closest boat is the Cambridge Punter in centre. Too far?**						
USER	**Yes, it is. How about a museum?**						
Error Modality	Slot	Gold	Prediction	Open	Num.	Temp.	Cat.
Spurious	(Attraction, Name)	`n/a`	cambridge punter	8.4	22.3	47.7	16.0
Ignored	(Hotel, Internet)	yes	`n/a`	65.3	53.5	19.9	76.8
Wrong Value	(Attraction, Type)	museum	boat	26.3	24.2	32.4	7.2

we perform zero-shot experiments using each representation. For tuple-based questions, our model takes as slot query a synonym of the slot name (e.g. Taxi 'leaveAt' → 'Pick Up Time') instead of a full question (e.g. 'What time does the user want the taxi to pick them up¿). Full question models achieved 6% higher per-domain JGA compared to slot-tuple models, supporting the notion that slot-tuple models memorize slot names rather than understanding their meaning and thus do not generalize well in real-world settings. Using full questions, our model (Table 3.4) achieves state-of-the-art performance in zero-shot settings.

3.6.3 Error Modalities

To analyze our model, we follow Gao et al. [55] and categorize DST errors in three modalities: 1) the model predicts a *spurious* value for an irrelevant slot; 2) the model *ignores* a relevant slot; and 3) the model correctly infers the presence of a slot but predicts a *wrong value*. Table 3.5 shows examples of each type of error for a sample conversation, and what proportion of errors they make up in each slot category for our **GPT2-DST** model in a zero-shot setting. Temporal slots are least likely to be *ignored* by our model, as verbatim `HH:MM` values are easily identifiable in a conversation. However, it is difficult to distinguish between closely related unseen temporal slots like 'leaveAt' and 'arriveBy'. Values for categorical, numeric, and open-valued slots on the other hand can comprise common (non-slot) phrases used in conversation, and thus it is easy for our model to ignore such slot references.

We also examine the source of dialog slots: *users* explicitly express the majority (79.5%) of slot values, while a minority are either derived via user reactions to *system* suggestions (9.7%) or *implicitly* valued (10.8%)—not present verbatim in a conversation. However, our errors are distributed evenly between *user*, *system*, and *implicit* sourced slots—suggesting that it is challenging for our model to track dialog states that are updated reactively via user feedback. We thus see a future opportunity to improve DST models by emphasizing multi-hop reasoning and common-sense inference.

3.6.4 Predicting Carried Over Slots

Long-range dependencies and slot values carried over from early turns are particularly important to model for accurate DST in long conversations [99]. We observe this in the zero-shot setting: our model is able to predict all slots accurately for 61% of conversation first-turns, dropping to 46% after one turn, and 5.7% after seven turns (the average conversation duration). We implement an oracle module to discard predictions when a dialog state does not need updating, obtaining an upper bound for DST improvements due to carry-over prediction. With this oracle, we see an average 5-point improvement in JGA across domain, indicating that carry-over prediction can greatly benefit our model. State-of-the-art models for fully supervised DST often rely on explicitly processing previous dialog states—via slot-value graphs [250, 25] or as a separate input to the model at each turn [70, 94]. In our framework we can target slot carry-over by training a model to predict a `carried over` token in place of the true slot value whenever a slot value does not need updating at the current turn (**+ Carryover**). At inference time, we replace predicted carry-over tokens with the slot's last predicted value.

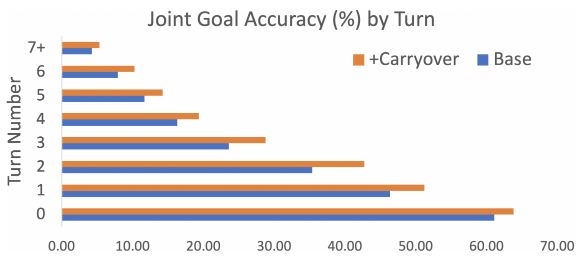
Our carry-over implementation improved JGA for all domains (Table 3.6a) by an average of 3.14%, and improved JGA across all context lengths—with the largest improvements (+7%) at the second and third turn of a conversation (Figure 3.6b). The `carried over` token allows

Table 3.6. Comparing previous state of the art, GPT2-DST, and the additive effect of carry-over prediction and transfer learning on per-domain zero-shot DST JGA (a). These two modifications to GPT2-DST both improve generalization in both small and medium models. In (b) we show the per-turn JGA of GPT2-DST with and without carry-over prediction.

(a) Zero-shot DST JGA

	R	H	A	T	X
Prior SOTA	16.5	19.8	22.6	22.8	59.5
GPT2-DST (s)	21.1	18.5	23.7	24.3	59.1
+ Carryover	24.0	19.9	28.5	30.8	59.3
+ DSTC8	24.7	22.9	34.3	38.6	59.7
GPT2-DST	26.2	24.4	31.3	29.1	59.7
+ CO, DSTC8	**27.7**	**24.9**	**42.4**	**41.1**	**60.3**

(b) Per-turn JGA



our model to hedge against low confidence slots, falling back to predictions from previous turns where the target slot may be directly mentioned. This helps reduce the *wrong value* error rate by an average of 31% across each domain. Our model can also propagate null values with carry-over, reducing spurious predictions by an average of 36% across domains. However, we also observe our carry-over model propagating 78% of its errors from previous turns, suggesting that further improvements can result via accurately predicting slot updates.

3.6.5 Transfer Learning for Generalization

Our framework is ontology-agnostic and thus easily supports transfer learning without modifying the architecture by simply writing natural language questions for additional slots. Gao et al. [55] found that intermediate fine-tuning of RoBERTa-Large [126] on passage-based QA tasks [50] improved zero-shot DST performance. In preliminary experiments, we found no significant impact from intermediate fine-tuning on the SQuAD v2.0 [171] passage-based QA dataset. However, we observe significant improvements when training with joint, non-curriculum learning [144, 170]—augmenting our training data with an equal number of examples sampled from DSTC8, taking care to remove data from the held-out domain in both MultiWOZ and DSTC8.

Our framework allows for easy joint optimization with carry-over and transfer learning: by training new models on MultiWOZ 2.1 augmented with DSTC8 (**+ DSTC8**) we gain a further average 3.5-point improvement in per-domain JGA (Table 3.6a). On average, our model makes 29% fewer *spurious* errors, and 6.9% fewer errors in open-valued slots, suggesting that our model scales well with additional training data with semantically distinct slot types and values. Our model also makes 9.7% fewer errors on categorical slots and 63% fewer mistakes where it assigns the value of one categorical slot to another, despite being unable to observe the set of possible categorical options—suggesting that exposure to more diverse categorical slots allows our model to better understand and distinguish between such slots. While temporal slots comprise only 17% of MultiWOZ and 10% of DSTC8 slots, these additional examples seem to help our model better disambiguate temporal references and make 32% fewer errors in such slots.

By applying both carry-over and transfer learning to our largest model, we observe further improvements in zero-shot JGA for all domains—averaging 5.1 points better than GPT2-DST, for an average gain of 11% JGA over previous state-of-the-art across domains (Table 3.6a).

3.7 Qualitative Analysis

We manually reviewed 300 errors made by our GPT2-medium CLMQ model in the zero-shot setting—annotating 20 errors from each modality (spurious, ignored, wrong value) from each domain with the gold label quality and perceived cause of error totaling 300 annotated examples. As widely observed in recent DST work [250, 99], a significant proportion of DST errors on MultiWOZ are unavoidable—caused by annotation errors. While version 2.1 corrected some of these, annotation errors and inconsistencies remain responsible for 30% of sampled errors—in particular, in 10% of errors the original annotator did not record reactive preferences while in 5% of errors the original annotator did. These inconsistencies can hurt our model's ability to infer reactive and implied requirements and preferences.

We are also particularly interested in *slot transfers*—when our model mistakenly predicts one slot's value for a different slot, comprising 36% of our manually reviewed errors. In the Taxi and Hotel domains, our model transfers slots from the same domain over 75% of the time, with most swaps occurs between same-category slots (e.g. temporal slots like Taxi 'LeaveAt' and 'ArriveBy'). Slots in these domains are closely semantically related, with values that can fit any slot of that category (e.g. 13:10 vs. 15:15). While a human can easily infer that the earlier of two times must be departure and the later arrival, our model has no inherent understanding of temporal mechanics or numeracy [213]. In future work, we will explore learning such knowledge directly via hierarchical softmax output distributions to distinguish between output modalities [195], and fine-tuning our model with contrastive losses to learn to rank numerals and times [76].

For Restaurant, Attraction, and Train, our model tends to swap slot values with those from other domains in the conversation. This is often due to semantically similar slots whose values, at first glance, may not be obviously identifiable as such (e.g. 'Bridge' or 'The Place'). Kumar et al. [99] similarly observe a particularly high incidence of slot transfers between different-domain 'Name' slots. Other such slots include price ranges and numbers of guests. We have seen that data augmentation with DSTC8 can improve our model's ability to disambiguate such slots—this suggests that we could further improve our model by exposing it to in-domain, conversational reading comprehension data.

While no such dataset currently exists, in future work we aim to explore using question generation [41] and paraphrasing [205] models to perform in-domain data augmentation, creating reading comprehension questions for task-oriented dialogs that targeting entities and relations not covered by an ontology. We also wish to explore methods for generating general reading comprehension questions for out-of-domain conversations [187] to improve our model's domain adaptation ability.

Table 3.7. Unique documents/entities/types and number of mentions in each split of **WikiWiki**. *Test (New Entities)* comprises entities not seen in the training split.

	Documents	Uq. Entities	Uq. Types	# Mentions	Type References
Training	10 M	2.2 M	40.6 K	38.7 M	43.8 M
Test	5.0 K	14.1 K	4.0 K	19.3 K	21.5 K
Test (New Entities)	5.0 K	6.0 K	1.2 K	6.4 K	6.5 K

3.8 Type-Centric Multitask Modeling

We observed in Section 3.6.1 that the knowledge accrued from language model pre-training was crucial in enabling zero-shot generalization in DST. Inferring slot values for unseen slots is an analogous task to identifying entities and their *types* or categorizations. In the second part of this chapter, we will thus explore a method to instill type-centric knowledge in language models, demonstrating that models trained under our pre-training framework can achieve even greater gains in zero-shot DST performance.

3.8.1 WikiWiki Corpus

To train an entity- and type-aware language model, we build the **WikiWiki** dataset by combining Wikipedia articles with the Wikidata KG [212]. Wikipedia articles have been used to enrich corpora for dialog [39], coreference resolution [192], and QA [123]. KGs have been used for entity typing and relation extraction [177]. Yao et al. [239] use Wikipedia pages as context for relation triples mined from Wikidata.

We link articles, entities, and types as in Figure 3.2: like Wu et al. [228], we take Wikipedia hyperlinks as links between *entities* (target page) and their *mentions* (link text); we link pages to Wikidata nodes via ID; and for each node we extract types T from Wikidata where $t \in T$ is an *instance* / *subclass* of the node (discarding entities with no types).[2] To address sparsity of hyperlinks, we follow Yao et al. [239] and use spaCy to identify additional entities.

[2]All humans on Wikidata are an instance of 'human'; we thus use the 'occupation' relation to determine their types.

We sample 10M articles for training, with two disjoint 5K-article splits for evaluation, containing seen and unseen (New Ent) entities respectively (Table 3.7). The ontology of Wikidata types forms a directed acyclic graph with 41K type nodes applying to 2.2M entities. Previous entity typing datasets rely on annotations from small groups of crowd-workers and include a small type ontology in the hundreds [121] and/or sacrifice label accuracy [29]. We instead rely on the cumulative, cross-checked annotations from tens of thousands of active Wikidata users.

Entities in Wikidata on average are assigned 1.28 types; for entities with multiple types, not all types are necessarily relevant to a context. For example, take the following passage: *"Obama was elected to the Illinois Senate in 1996, succeeding Democratic State Senator Alice Palmer from Illinois's 13th District, which, at that time, spanned Chicago South Side neighborhoods from Hyde Park–Kenwood south to South Shore and west to Chicago Lawn."*

While Wikidata entities may have 5+ types, many are not directly relevant to a context. For example, while Barack Obama has types including *Politician*, *Jurist*, *Political Writer*, *Community Organizer*, and *Podcaster*, the latter is not relevant to the context. To teach our models to infer types relevant to the context at hand, in pre-training data we take only types that are shared between Barack Obama and other entities in the document (e.g. Alice Palmer—Politician). We have made the WikiWiki dataset publicly available on Github.[3]

3.8.2 Pre-training Tasks

To instill type-centric knowledge from WikiWiki, we train our models to answer four types of knowledge-based questions conditioned on a passage from Wikipedia (examples in Table 3.8). In *entity/type discovery*, the model is tasked to recover all surface forms (mentions) that reference an entity, along with their types—this is analogous to simultaneous entity recognition and typing. *Entity typing* consists of assigning types to an entity of interest. For *entity recognition*, we follow Cao et al. [21] by training our model to respond with an entity's full name and type when queried with a surface form. In *slot filling* we ask our model to return all entities mentioned in the passage

[3]https://github.com/amazon-research/wikiwiki-dataset/

Table 3.8. In pre-training, the model reads a Wikipedia article and answers questions from four tasks involving entities and types. It must generate answers containing terms not found verbatim in the text. Surface forms (mentions) in **green**, entities in **red**, and types in **blue**.

Context: These included carbon dioxide by burning diamond, and mercuric oxide by heating mercury. This type of experiment contributed to the discovery of "dephlogisticated air" by Priestley, which became better known as oxygen, following Lavoisier's investigations.
Entity/Type Discovery (20%): List all concepts and types mentioned here. **Answer:** Priestley (chemist), Lavoisier (chemist), mercuric oxide (chemical compound), mercury (chemical element), and dephlogisticated air (superseded scientific theory)
Entity Typing (30%): What is dephlogisticated air an example of? **Answer:** superseded scientific theory
Entity Recognition (20%): What does Priestley refer to? **Answer:** Joseph Priestley (chemist)
Slot Filling (30%): Which chemists are mentioned here? **Answer:** Joseph Priestley and Antoine Lavoisier

belonging to a certain type. For multi-type entities, we use a subset of relevant types given other entities in the context. We treat QA as a universal format for diverse NLU tasks [144], and adopt the framework of Raffel et al. [170] to treat each of our tasks as text-to-text generative modeling. We create 50M questions for pre-training.

3.8.3 Type-Centric Model Architecture

We use an encoder-decoder [199] model initialized from BART—a Transformer [210] language model pre-trained via de-noising autoencoding [108]. Our model generates an answer a as a text sequence given a document D of length t_d and question q. The document is encoded via the encoder—consisting of l Transformer layers of hidden dimensionality h, each applying 16-headed self-attention—to produce $z := \text{Enc}(D) \in \mathbb{R}^{t_d \times h}$.

We train the model to perform QA via conditional language modeling. Instead of concatenating the question with the context in encoder input [120], the decoder generates a sequence consisting of the question and answer: $x = [q; a]$. We can thus cache the document encoding at inference to answer multiple questions. At training time we perform next-token prediction, calculating cross-entropy loss by maximizing the log likelihood of the question and answer conditioned on the document:

$$P(q, a|D) = \prod_{t}^{T} P(x_t|x_{<t}, D)$$

We assess the impact of our pre-training on Base (l=12, h=768) and Large (l=24, h=1024) models.

3.9 Type-Aware Model Experiments

We demonstrate the effectiveness of our pre-training on two tasks that require type understanding: fine-grained entity typing, and zero-shot domain generalization in dialog state tracking (DST).

3.9.1 Fine-Grained Entity Typing

Ultra-Fine Entity Typing

Our method improves generalization in type-adjacent tasks; we next aim to infer entity types in unseen documents. In preliminary experiments on the UltraFine dataset with 11K types [29], our models under-perform SOTA (24.0 vs. 49.1 F1). Manual inspection of gold labels reveals two main causes for error: 1) inaccurate labels—e.g. “rare plants” as type “bird”; and 2) inconsistent usage of gold labels: different spellings (*organization* / *organisation*) or synonyms (*car* / *automobile*) are treated as distinct and often do not collocate. This suggests that label noise in UltraFine may make it unsuitable for assessing granular, hierarchical type knowledge.

We examine these annotation errors via **human evaluation**, presenting crowd-workers with 200 contexts from UltraFine (10% of the test set). Only 68% of gold type labels were judged accurate, and 21% inaccurate. We compare gold labels against zero-shot predictions from our model in a second trial with 200 pairs. Judges preferred our predictions 51% of the time compared to 29% for gold. We observed moderate inter-annotator agreement of κ=0.4044 [51]. This suggests that our models can accurately infer types, but current benchmarks do not suitably

Table 3.9. P/R/F1 of pred. vs. gold types on WikiWiki Test (seen) and Test New Ent (unseen entities) splits.

Evaluation Set	Documents	Entities	Types	Model	Precision	Recall	F1 Score
Seen	5.0 K	14.1 K	4.0 K	RoBERTa	62.35	59.38	60.82
				Ours	**78.13**	**72.39**	**75.15**
Unseen	5.0 K	6.0 K	1.2 K	RoBERTa	48.88	47.96	48.41
				Ours	**66.65**	**63.71**	**65.14**

measure typing quality.

Entity Typing on WikiWiki

We turn to WikiWiki to evaluate fine-grained entity typing, leveraging type labels verified by active users of Wikidata. To verify the accuracy of ground-truth type labels in the WikiWiki test set, we asked human evaluators to judge the accuracy of 443 type labels from 200 randomly sampled contexts. We confirm that WikiWiki is a high-quality benchmark for entity typing, with 85% type precision assessed by human judges (compared to 68% for UltraFine).

We found that multi-label classifiers built on RoBERTa [126] that perform well on UltraFine require significant hyper-parameter tuning to output non-trivial predictions to classify our large and sparse (41K) type ontology. To perform entity typing with our model, we generate comma-delimited text sequences of types [235]. This allows our models to infer and generate novel types while classifiers remain restricted to the training ontology. We confirm that our pre-training helps models better infer types for both seen (+14.3 F1) and unseen entities (+16.7 F1) in new contexts compared to classifiers (Table 3.9).

To investigate if our model can discover novel types, we perform another **human evaluation** over 557 such predictions from 300 contexts, with inter-annotator agreement of κ=0.4086. Our model accurately extrapolates its type knowledge beyond the training ontology—we observe 73.3% precision when inferring new types (compared to 74.5% precision for seen types), demonstrating that our pre-training enables models to reason about types beyond simple memorization. Our model discovers complex and specific scientific types, correctly proposing

Table 3.10. Zero-shot domain adaptation JGA (%) on MultiWOZ 2.1 test set on the Restaurant, Hotel, Attraction, Train, and Taxi domains. We achieve SOTA results on all domains by significant margins, with an average absolute improvement of 9.7 points and relative improvement of 30.6% across domains.

	# Params	Restaurant	Hotel	Attraction	Train	Taxi
TRADE	90M	12.6	14.2	20.1	22.4	59.2
MA-DST	90M	13.6	16.3	22.5	22.8	59.3
SUMBT	355M	16.5	19.8	22.6	22.5	59.5
GPT2-DST	355M	26.2	24.4	31.3	29.1	59.6
BART	139M	27.9	31.9	38.4	34.3	70.5
Ours (Base)	139M	*40.4*	*36.5*	*39.8*	*36.1*	*70.9*
Ours (Large)	406M	**46.7**	**38.8**	**49.8**	**37.7**	**72.1**

that anorthosite (an aluminum silicate rock) is a *metallurgical rock*[4] and that speckled tortoises are *monotrophs*.[5] This reflects the robust taxonomy of types in scientific disciplines. Our model also proposes granular categories of events, and is judged to correctly type the 2015 Tour of Taiwan as an instance of the *Tour de Taiwan* cycling race. In the future, we seek methods to automatically assess the factual accuracy of new types.

3.10 Zero-Shot DST

As discussed in Section 3.3, in many real-world settings, DST models must be able to predict new slot values (i.e. new entities that are not present in the training corpus) and new slot types (e.g. requirements for applications in new domains). We use the same question answering setup for our type-aware model as for GPT2-DST. Our method is complementary to systems for creating synthetic in-domain dialogs [95].

As seen in Table 3.10, our type-centric pre-training allows a model to answer questions about unseen slots. BART-base itself achieves SOTA JGA across all domains, and our pre-training significantly increases the gain to 10.6% absolute / 34.8% relative JGA—despite only using one-third of the parameters. Our Large model achieves 14.9% absolute and 49.4% relative gain in

[4]rocks containing metallic compounds and properties

[5]has diet comprising one type of food [74]

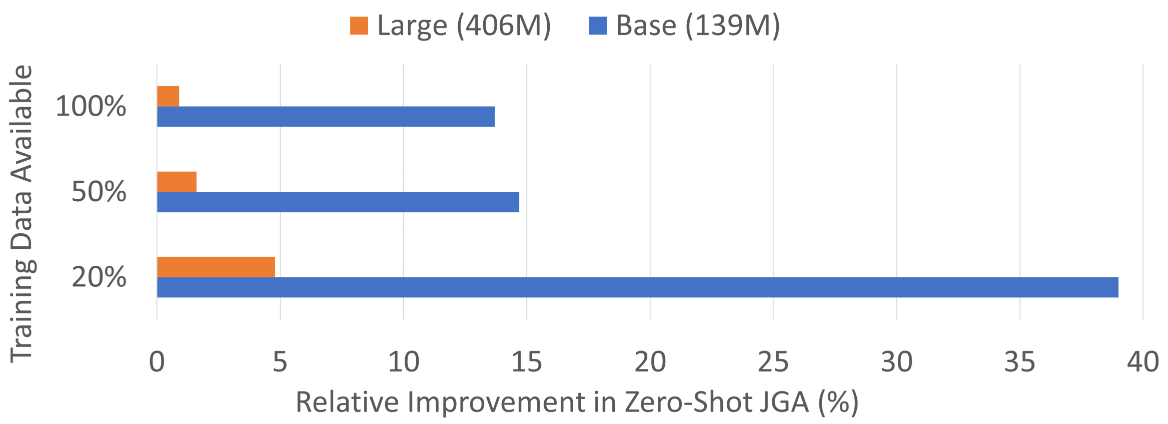


Figure 3.5. Relative gain (%) in JGA for models trained on WikiWiki vs standard BART pre-training. Our method helps more in low-data regimes and for smaller models.

JGA compared to previous SOTA. The most significant gains come in the Hotel and Restaurant domains, which contain the most categorical slots that resemble types (e.g. cuisine, hotel type).

In Figure 3.5 we compare our models against same-size BART models at different levels of training data availability to demonstrate the additive utility of our method. Our method is particularly helpful with less fine-tuning data (low-data regimes), with average gains of 39% for small models and 4.8% for large models at 20% data availability. Gains are magnified for smaller models, affirming that our method can effectively instill type knowledge in lightweight language models.

3.11 Conclusion

This chapter proposes a conditional language modeling approach to multi-domain DST posed as a generative question answering task. By leveraging natural language questions as state queries, our model can generalize to unseen domains, slots, and values via its understanding of language. Our model achieves state-of-the-art zero-shot results on the MultiWOZ 2.1 dataset with average per-domain absolute improvements of 5.9% joint accuracy. We also demonstrate that our framework is easily extensible to support transfer learning and learning slot carry-over. We next 1) propose a text-to-text pre-training scheme to instill type knowledge in language models via QA and 2) release the **WikiWiki** dataset built from Wikipedia articles and the Wikidata KG. We show that WikiWiki is larger-scale and more accurate than existing fine-grained type recognition datasets. We demonstrate that our type-centric pre-training framework allows us to train language models that can better generalize to unseen domains, entities, and types—which in turn lead to improved model performance on downstream tasks like dialog state tracking (achieving SOTA results on zero-shot DST with average gains of 14.9% joint accuracy above our previous best model). Our models can extrapolate type knowledge and infer novel types that humans judge to be useful and precise. As the body of human knowledge grows, we see an opportunity to use life-long learning [157] on news and publications to expand and model the taxonomy of knowledge. This would permit even more effective generalization for preference elicitation and personalization for open-domain assistive agents.

3.12 Acknowledgements

This chapter contains material as it appears in and adapted from the following papers: 1) Conference of the European Chapter of the Association for Computational Linguistics, 2021 ("Zero-shot Generalization in Dialog State Tracking through Generative Question Answering" Shuyang Li, Jin Cao, Mukund Sridhar, Henghui Zhu, Shang-Wen Li, Wael Hamza, Julian McAuley); 2) Findings of Annual Conference of the North American Chapter of the Association for Computational Linguistics, 2022 ("Instilling Type Knowledge in Language Models via Multi-Task QA" Shuyang Li, Mukund Sridhar, Chandana Satya Prakash, Jin Cao, Wael Hamza, Julian McAuley). The dissertation author was the primary investigator and author of both papers.

Chapter 4

Transcript-Free Conversational Recommendation with Rationales

4.1 Introduction

Traditional recommender systems often give static suggestions, affording users no way to meaningfully express their preferences and feedback. Conversational recommendation allows users to interact with agents and suggestions, increasing their willingness to trust and accept recommendations [164]. Techniques for conversational recommendation are based on the iterative *paradigm* of conversation: how an agent can explain their suggestions and how users can give feedback. Existing approaches for conversational recommendation are either unable to explain their suggestions, cannot handle multiple rounds of dialog, or rely on expensive conversational transcripts to train. **In this chapter,** our goal is to build a simple yet effective transcript-free framework that can train conversational recommendation agents that can 1) explain their rationales for recommendations, 2) incorporate user feedback to adapt its behavior to users' needs, and 3) effectively improve recommendation quality over multiple rounds of conversation.

Recent work has explored conversational recommendation through dialog agents trained to ask the user questions in free-form dialog [220]. Such models require large training corpora comprising transcripts from crowd-sourced recommendation games [83]. To create high-quality training data, crowd-workers must be knowledgeable about many items in the target domain—this expertise requirement limits data collection to a few common domains like movies. It is thus

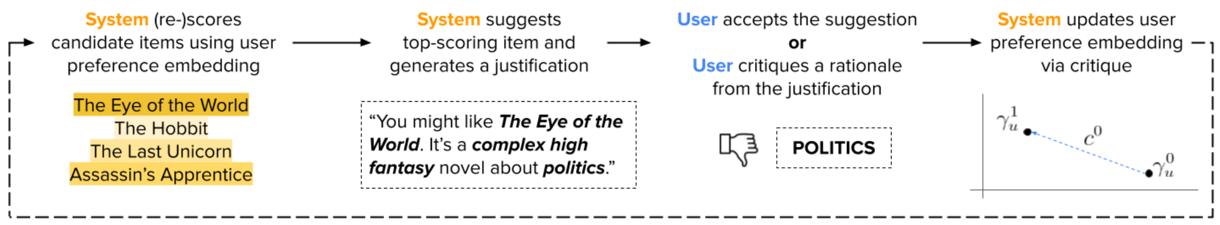

System (re-)scores candidate items using user preference embedding
The Eye of the World
The Hobbit
The Last Unicorn
Assassin's Apprentice
System suggests top-scoring item and generates a justification
"You might like The Eye of the World. It's a complex high fantasy novel about politics."
User accepts the suggestion
or
User critiques a rationale from the justification
POLITICS
System updates user preference embedding via critique
γ_u^1
c^0
γ_u^0

Figure 4.1. In our conversational recommendation workflow, the system scores candidates and generates a justification for the top item. If the user critiques a rationale, the system uses the critique to update the latent user representation.

difficult to scale dialog-based recommenders to domains where users have specific preferences about subjective rationales but no dialog transcripts exist (e.g. food and literature).

We address this challenge of data scarcity by proposing a framework for training conversational recommender systems based on conversational critiquing and self-supervised bot-play. Our approach reflects an *realistic interactive paradigm* where the agent suggests items and explains their rationale, while the user specifies their preferences via specific feedback to guide the next turn's suggestions [249]. Our framework does not rely on supervised dialog examples and can be applied to *any* setting where product reviews or opinionated text can be harvested.

We propose a framework comprising two parts: First, we learn to jointly recommend items and generate justifications based on subjective rationales, leveraging ideas from conversational critiquing systems [227] trained via next-item recommendation. We then fine-tune our model for multi-turn recommendation via multiple turns of bot-play in a recommendation game based on natural-text product reviews and simulated critiques.

Our framework is model-agnostic—we apply our method to two different underlying recommendation architectures [182, 175] of differing sizes and evaluate our models on three large real-world recommendation datasets with user reviews but no dialog transcripts. Our method can provide more useful explanations and better adapts to user feedback compared to state-of-the-art (SOTA) conversational recommender systems—users interacting with our rationales reach their goal items faster and with greater success. We conduct a study with real users, showing that our models can effectively help users find desired items in real time, even in a cold-start setting.

We summarize our main contributions as follows: 1) We present a framework for training conversational recommender systems using bot-play on historical user reviews, without the need for large collections of human dialogs; 2) We apply our framework to two popular recommendation models (**BPR-Bot** and **PLRec-Bot**), with each showing superior or competitive performance in comparison to SOTA recommendation and critiquing methods; 3) We demonstrate through human evaluation and user studies that models trained with our bot-play framework are more useful, informative, knowledgeable, and adaptive compared to SOTA baselines.

4.2 Related Work

4.2.1 Justifying Recommendations

Users prefer recommendations that they perceive to be transparent or justified [193]. Some early recommender systems presented the same attributes of suggested items to all users [211]. Another line of work attempts to generate natural language explanations of recommendations. McAuley et al. [142] mine key attributes from textual reviews via topic extraction. These attributes can be expanded into explanatory sentences via template-filling [246] or recurrent language models [152]. In this chapter, we allow the user to provide feedback about specific rationales mentioned across natural language product reviews in large recommendation datasets.

4.2.2 Conversational Critiquing

Critiquing systems allow users to incrementally construct preferences, mimicking how humans refine their preferences based on conversation context [206]. Early critiquing methods treated user feedback as hard constraints to shrink the search space [18]. Wu et al. [227] introduced a critiquing model with justifications comprising natural language attributes mined from user reviews—with which users can then interact. Antognini et al. [4] provide a single-sentence explanation alongside a set of rationales, requiring users to interact only with the rationale set. Luo

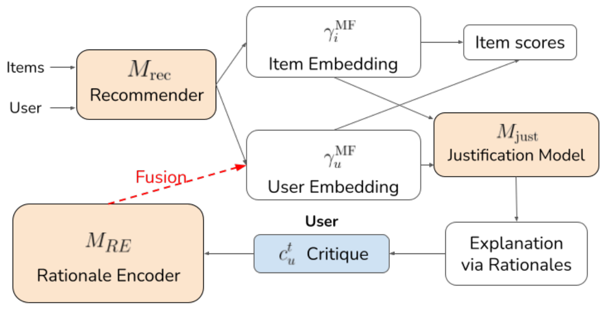
Table 4.1. Critiquing systems (top) are not equipped for multi-turn interactions. Q & A systems (middle) ask the user to build a list of search criteria but do not provide rationales for recommended items. Dialog agents (bottom) learn multi-turn behavior via large corpora of domain-specific transcripts. Our framework allows us to train conversational recommenders without costly transcript data.

Paradigm	Model	Justifies Suggestions	Multi-Turn Conversations	Transcript-Free
Critiquing	LLC [132]	✗	✗	✓
	CE-VAE [133]	✓	✗	✓
	M&M VAE [3]	✓	✗	✓
Q & A	SAUR [247]	✗	✓	✓
	EAR [104]	✗	✓	✓
	SCPR [105]	✗	✓	✓
Dialog Agents	Li et al. [113]	✗	✓	✗
	Kang et al. [83]	✓	✓	✗
	Zhou et al. [249]	✓	✓	✗
	Ours	✓	✓	✓

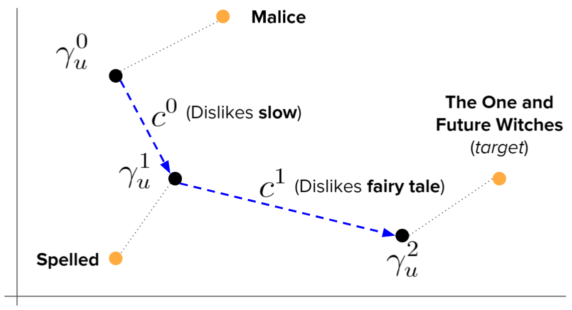
et al. [133] use a variational auto-encoder (VAE) [97] for joint recommendation and justification, learning a bi-directional mapping function between latent user and rationale representations. Current critiquing techniques are either trained only for next-item recommendation, or to handle a single turn of critiquing [3], and struggle to incorporate feedback in multi-turn settings. We adopt techniques for encoding user feedback from critiquing systems [132], but we introduce a multi-step, model-agnostic bot-play method to explicitly train our models for multi-turn conversational recommendation.

4.2.3 Dialog Agents for Recommendation

We view recommenders as domain experts who can elicit preferences from human customers and suggest appropriate items over the course of a session [19]. A recent line of work formulates conversational recommendation as goal-oriented dialog: at each turn, the user is either a) asked if they prefer a specified attribute; or b) recommended an item [30, 247]. Other question-answering models use reinforcement learning to dialog policies for when to ask users



(a) Model Architecture


Malice
γ_u^0
c^0 (Dislikes **slow**)
The One and Future Witches (*target*)
γ_u^1
c^1 (Dislikes **fairy tale**)
Spelled
γ_u^2

(b) Latent Critiquing Process

Figure 4.2. (a) Given a user, items, and rationale critique vector, our model encodes the critique $M_{\text{RE}}(c_u^t)$ and fuses it with the user embedding γ_u^{MF} via critiquing function f_{crit}. The fused user (γ_u) and item (γ_i) representations are then used to predict the justification and score items. An example is shown in (b), where user feedback about the rationales *slow* (c^0) and *fairy tale* (c^1) modify our prior latent user preference vector to bring it closer to the target item ("The One and Future Witches").

about attributes, updating a cumulative belief state of item attributes [104, 105]. These models ask templated questions and surface recommendations from an open candidate pool without explaining their reasoning to the user.

Another line of research treats conversational recommenders as free-text dialog agents that interact with users via natural language utterances. Bot-play has been explored as a way to train such dialog agents [113, 83], which requires models to be trained and fine-tuned using existing dialog transcripts. Such agents are thus limited to domains where crowd-sourced workers can accurately play the roles of expert and seeker to collect data via Wizard-of-Oz setups [35]. By allowing users to critique natural text rationales of a suggested item, our framework for conversational recommendation allows for multi-turn recommenders that can be trained using only product review texts—which are available in a wide range of domains. In Table 4.1 we compare our approach to recent frameworks for critiquing and dialog agents for conversational recommendation.

4.3 Model

Our model comprises (Figure 4.2):

1. a recommender model M_{rec} that ranks items based on their suitability for a user;
2. a justification module M_{just} that predicts rationales for a given recommendation; and
3. an interactive critiquing function f_{crit} that allows users to edit a rationale and modifies the user representation to recommend a different item on the next turn.

We support multi-step critiquing (Figure 4.2): at each turn a user may indicate which rationales they dislike about the current suggestions via a critique c^t. The critiquing function then modifies the latent user representation γ_u via the critique to bring it closer to the target item.

4.3.1 Recommender System

Our method can be applied to any recommender that learns user and item representations. We show its effectiveness with two popular methods:

Bayesian Personalized Ranking (BPR) [175] is a matrix factorization recommender system that aim to decompose the interaction matrix $\mathbf{R} \in \mathbb{R}^{|U|\times|I|}$ into user and item representations [98]. BPR optimizes a ranked list of items given implicit feedback (binary interactions between users and items). Scores are computed via inner product of h-dimensional user and item embeddings: $\hat{x}_{u,i} = \langle \gamma_u^{\text{MF}}, \gamma_i^{\text{MF}} \rangle$. At training time, the model is given a user u, observed item i and unobserved item j. We maximize the likelihood that the user prefers the observed item:

$$\mathcal{L}_R = P(i >_u j|\Theta) = \sigma(\hat{x}_{u,i} - \hat{x}_{u,j})$$

where σ represents the sigmoid function $\frac{1}{1+e^{-x}}$.

Projected Linear Recommendation (PLRec) is an SVD-based method to learn low-rank user/item representations via linear regression [182]. The PLRec objective minimizes:

$$\arg\min_W \sum_u \| r_u - r_u V W^T \|_2^2 + \Omega(W)$$

where V is a fixed matrix obtained by taking a low-rank SVD approximation of $\mathbf{R}$ such that $\mathbf{R} = U\Sigma V^T$, and W is a learned embedding. We obtain an h-dimensional embeddings for users ($\gamma_u^{\text{MF}} = r_u V$) and items ($\gamma_i^{\text{MF}} = W_i$).

4.3.2 Justification Module

Our justification model (rationale prediction head) consists of a fully connected network with two h-dimensional hidden layers predicting a score $s_{u,i,a}$ for each natural language rationale a. This model takes the sum of user and item embeddings as input. At training time, we incorporate a rationale prediction loss $\mathcal{L}_A$ by computing the binary cross entropy (BCE) for each rationale given the likelihood the user cares about the rationale:

$$\mathcal{L}_A = -\frac{1}{|A|}\sum_{a=0}^{|A|} \mathbf{k}_{i,a}^I \cdot \log p_{u,i,a} + (1 - \mathbf{k}_{i,a}^I) \cdot \log(1 - p_{u,i,a})$$

At inference time, we again compute the likelihood for each rationale $p_{u,i,a} = \sigma(s_{u,i,a})$ and sample from the Bernoulli distribution with $p_{u,i,a}$ to determine which rationales a appear in the justification.

4.3.3 Critiquing Function

We posit that the user's latent representation is partially explained by their written reviews. We thus learn a rationale encoder M_{RE}—a linear projection from the rationale space to the user preference space: $M_{\text{RE}}(c_u^t) = W^T c_u^t + b$, where $c_u^t \in \mathbb{Z}^{|K|}$ is the critique vector representing the strength of a user's preference for each rationale. We fuse this rationale encoding with the latent user embedding from M_{rec} to form the final user preference vector: $\gamma_u = f(\gamma_u^{\text{MF}}, M_{\text{RE}}(c_u^t))$. For both models, we fuse via the element-wise mean of the two vectors: $f(a, b) = \frac{a+b}{2}$. In training, the rationale encoder takes in the user's rationale history: $c_u^t = \mathbf{k}_u^U$.

Algorithm 2: Bot play framework for fine-tuning conversational recommenders.

```
Recommender and Justifier $M_{\text{rec}}, M_{\text{just}}$;
Critique fusion function $f_{\text{crit}}$;
Seeker model $M_{\text{seeker}}$;
for each user $u$ do                      > Fine-tune across users in training set
    for goal item $g \in I_u^+$ do        > Sample goal item from reviewed items
        initialize $\gamma_u^1$ from $M_{\text{rec}}$, $\mathcal{L} = 0$;
        for turn $t \in range(1, T)$ do   > Simulate up $T$ turns of feedback
            $\hat{x}_{u,i}^t = M_{\text{rec}}(\gamma_u^t, i) \; \forall \; i \in I$;
            $\mathcal{L} \leftarrow \mathcal{L} + \delta^t \cdot \mathcal{L}_{\text{CE}}(g, \hat{x}_{u,i}^t)$;
            $\hat{i}^t = \arg\max_i \hat{x}_{u,i}^t$;
            if $\hat{i}^t = g$ then       > Terminate if goal item recommended.
                break with success
            $\hat{k}_{u,\hat{i}^t} = M_{\text{just}}(\gamma_u^t, \gamma_{\hat{i}^t})$;   > Generate rationales
            simulate user critique using $M_{\text{seeker}}$: $c_u^t$;
            $\gamma_u^{t+1} \leftarrow f_{\text{crit}}(\gamma_u^t, c_u^t)$;   > Update user latent representation
return fine-tuned agent
```

Critiquing with Our Models

To perform conversational critiquing with a model trained using our framework, we adapt the latent critiquing formulation from Luo et al. [132], as shown in Figure 4.1. At each turn t of a session for user u, the system assigns scores $\hat{x}_{u,i}^t$ for all candidate items i, and presents the user with the highest scoring item $\hat{i}$. The system also justifies its prediction with a set of predicted rationales $\hat{k}_{u,i}^t$. The user may either accept the recommended item (ending the session) or critique a rationale from the justification: $a \in \{a | \hat{k}_{u,i,a} = 1\}$.

Given a user critique, the system modifies the predicted scores for each item and presents the user with a new item and justification:

$$\hat{x}_{u,i}^{t+1} = M_{\text{rec}}(\hat{\gamma}_u^{t+1}, i)$$
$$\hat{k}_{u,i}^{t+1} = M_{\text{just}}(\hat{\gamma}_u^{t+1}, i)$$
$$\hat{\gamma}_u^{t+1} \leftarrow f_{\text{crit}}(\hat{\gamma}_u^t, c_u^t)$$

Effectively, a user critique modifies our prior for the user's preferences; we then re-rank the items presented to the user.

At inference time, we initialize the cumulative critique vector c_u^t with the user's rationale history ($c_u^0 = \mathbf{k}_u^U$). It is then updated via:

$$c_u^t = c_u^{t-1} - \max(\mathbf{k}_u^U, 1) \odot m_u^t; \quad c_u^0 = \mathbf{k}_u^U$$

where $\odot$ is element-wise multiplication. Here the critique should match the strength of a user's previous opinion of the rationale $\mathbf{k}_u^U$. Even if a user has not mentioned a rationale in their previous reviews, the max ensures a non-zero effect from each critique.

4.3.4 Training

To train our BPR-based model, we jointly optimize each component. Each training example comprises a user and observed / unobserved items. We predict scores for each item:

$$\hat{x}_{u,i} = \langle \gamma_u^{\mathrm{MF}} + M_{\mathrm{RE}}(\mathbf{k}_u^U), \gamma_i \rangle$$

We first compute the BPR loss (see Section 4.3.1) with the predicted observed / unobserved scores. We add the rationale prediction loss, scaled by a constant λ_{KP} to the ranking loss for our training objective: $\mathcal{L} = \lambda_{\mathrm{KP}}\mathcal{L}_A - \mathcal{L}_R$. We find empirically that $\lambda_{\mathrm{KP}} \in \{0.5, 1.0\}$ works well.

To train our PLRec-based model, we follow Luo et al. [132] and separately optimize M_{rec}, M_{just}, and M_{RE}. We optimize M_{RE} via the linear regression:

$$\arg\min_{W,b} \sum_u \| \gamma_u^{\mathrm{MF}} - M_{\mathrm{RE}}(\mathbf{k}_u^U) \|_2^2 + \Omega(W)$$

Finally, we optimize the rationale prediction (justification) loss $\mathcal{L}_A$ to train the justification head.

4.3.5 Learning to Critique via Bot Play

We propose a framework for critiquing via bot play that simulates user sessions when provided just a set of user reviews. We first pre-train our expert model (recommender, justifier, and rationale encoder). A seeker model is pre-trained via a simple prior: provided a target item and justification, it selects the most popular rationale present in the justification but not the target's historical rationales $\mathbf{k}_i^I$ to critique. For each training example (user and a goal item they have reviewed), we allow the expert and seeker models to converse with the goal of recommending the goal item. We fine-tune the expert by maximizing its reward (minimizing loss) in the bot-play game (Algorithm 2). We end the session after the goal item is recommended or a maximum session length of $T = 10$ turns is reached. We define the expert's loss to target both surfacing the correct recommendation and inferring the user's ground truth preferences per turn:

$$\mathcal{L}^{\text{expert}} = \sum_{t}^{T} \delta^{t-1} \cdot (\mathcal{L}_{\text{CE}}(g, \hat{x}_{u,i}^{t}) + \frac{1}{2}\mathcal{L}_A)$$

where δ is a discount factor to encourage successfully recommending the goal item at earlier turns, $\mathcal{L}_{\text{CE}}(g, \hat{x}_{u,i}^{t})$ is the cross-entropy loss between predicted scores and the goal item, and $\mathcal{L}_A$ is the binary cross-entropy rationale loss defined in Section 4.3.2. We find that a discount factor of $\delta = 0.9$ is effective for both BPR- and PLRec-based conversational recommenders.

4.4 Experimental Setting

We select hyperparameters for our initial models via AUC, and for bot-play fine-tuning via the success rate at 1 (SR@1) on the validation set. We train each model once, taking the median of three evaluation runs per experimental setting. For baseline models, we re-used the authors' code. All experiments were conducted on a machine with a 2.2GHz 40-core CPU, 132GB memory and one RTX 2080Ti GPU. We use PyTorch version 1.4.0 and optimize our models using the Rectified Adam [124] optimizer. Best hyperparameters for each base recommender

Table 4.2. Best hyperparameter settings for each base recommendation model. Linear critiquing methods (UAC, BAC, LLC-Score, LLC-Rank) use PLRec as a base model. BPR-Bot uses BPR as a base model, and PLRec-Bot uses PLRec as a base model.

Dataset	Model	h	LR	λ_{L2}	λ_{KP}	λ_c	β	Iterations	Epoch
Books	BPR [175]	20	0.001	0.01	0.5	–	–	–	200
	PLRec [182]	50	–	80	–	–	–	10	–
	CE-VAE [133]	100	0.0001	0.0001	0.01	0.01	0.001	–	300
Beer	BPR	20	0.001	0.01	0.5	–	–	–	200
	PLRec	50	–	80	–	–	–	10	–
	CE-VAE	100	0.0001	0.0001	0.01	0.01	0.001	–	300
Music	BPR	20	0.01	0.1	1.0	–	–	–	100
	PLRec	400	–	1000	–	–	–	10	–
	CE-VAE	200	0.0001	0.0001	0.001	0.001	0.001	–	600

Table 4.3. Dataset statistics, including number of unique rationales (R), sample subjective rationales from user reviews, and average unique rationales per user, item, and review.

	Users	Items	Rev.	Uq. R	Sample Subjective R	R/User	R/Item	R/Rev.
Books	13.9 K	7.6 K	655 K	75	Realistic, Strong Female	25.0	27.0	1.77
Beer	6.4 K	4.0 K	935 K	75	Citrus, Nutty, Bitter	54.6	60.2	7.39
Music	5.6 K	4.4 K	119 K	80	Techno, Catchy, Upbeat	16.5	20.0	2.54

system model are shown in Table 4.2. We perform hyperparameter search over a coarse sweep of: $h \in [2, 512]$, $LR \in [1e-5, 1e-2]$, $\lambda \in [1e-5, 1e-2]$. Model parameter sizes are a function of the hidden dimensionality h and number of items $|I|$ and users $|U|$, and is dominated by $h \cdot (|I| + |U|)$.

4.4.1 Datasets

We evaluate our models on three public real-world recommendation datasets with 100K+ reviews each: Goodreads Fantasy (Books) [214], BeerAdvocate (Beer) [142], and Amazon CDs & Vinyl (Music) [143]. We keep only reviews with positive ratings, setting thresholds of $t > 4.0$ for Beer and Music and $t > 3.5$ for Books. All reviews in these dataset are in English; we hope to extend our work to identify related rationales in multi-lingual reviews in the future. We partition each dataset into 50% training, 20% validation, and 30% test splits.

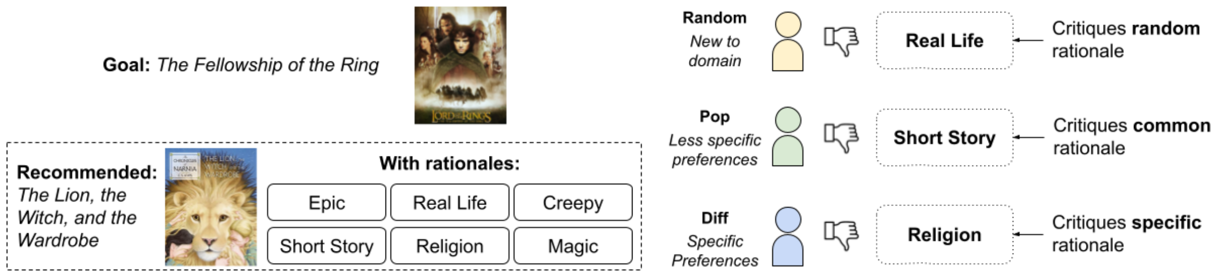

Goal: The Fellowship of the Ring
Recommended: The Lion, the Witch, and the Wardrobe
With rationales:
Epic
Real Life
Creepy
Short Story
Religion
Magic
Random
New to domain
Pop
Less specific preferences
Diff
Specific Preferences
Real Life
Short Story
Religion
Critiques random rationale
Critiques common rationale
Critiques specific rationale

Figure 4.3. Example of user behaviors after receiving a book recommendation with rationales. A new reader may *randomly* select a rationale to critique. Readers with less specific preferences may critique common / *popular* rationales. A knowledgeable reader with specific preferences will critique a specific *weakest* (most different from their target) rationale.

We follow the pipeline of Wu et al. [227] to extract subjective rationales (Table 4.3) from user reviews:

1. Extract high-frequency unigram and bigram noun- and adjective phrases;
2. Prune bigram keyphrases using a Pointwise Mutual Information (PMI) threshold, ensuring rationales are statistically unlikely to have randomly co-occurred; and
3. Represent reviews as sparse binary vectors indicating whether each rationale was expressed in the review.

These noun/adjective phrase rationales describe qualities ranging from taste for beers (e.g. citrus) and emotions for music (e.g. soulful) to perceived character qualities in books (e.g. strong female). Our framework is agnostic to the rationale format, and in future work we aim to extend our models to encode full sentences and utterances as critiques.

4.4.2 Multi-Step Critiquing

Following prior work on critiquing [132, 110], we simulate multi-step recommendation sessions to assess model performance. We simulate user sessions following Algorithm 2, with two main differences: 1) We randomly sample user u and their goal item g from the *test* set, and 2) We do not compute loss or update our model during a session. We set a maximum session limit of $T = 10$ turns. To evaluate how our models can help different types of users, we simulate each observation with three different critique selection strategies [110] as seen in Figure 4.3:

1. **Random**: Users who are new to the domain (e.g. new readers) tend to critique rationales at random;

2. **Pop**: Users with some domain knowledge and general preferences can correct more common rationales; and

3. **Diff**: Knowledgeable users with *specific* preferences will try to correct the *weakest* rationale.

In all settings, a user may only see any single item once and critique each rationale once per session.

4.4.3 Candidate Algorithms

Our method can apply to any base recommender system; here we train bot-play models based on BPR and PLRec—**BPR-Bot** and **PLRec-Bot** respectively. BPR-Bot is lightweight and much faster, while PLRec-Bot is similar in size to SOTA baseline models for conversational critiquing. We demonstrate in Section 4.5.1 that our framework is indeed model agnostic, and that BPR-Bot and PLRec-Bot both out-perform baselines.

Baseline methods

We assess linear critiquing baselines that co-embed critique and user representations [132], where f_{crit} is a weighted sum of the user preference vector γ_u and embeddings for each critiqued rationale. **UAC** uniformly averages γ_u and all critiqued rationale embeddings. **BAC** averages γ_u with the *average* of critiqued rationale embeddings. **LLC-Score** learns weights by maximizing the rating margin between items containing critiqued rationales and those without. Instead of directly optimizing the scoring margin, **LLC-Rank** [110] minimizes the number of ranking violations. These models cannot generate justifications; we binarize the historical rationale frequency vector for the item ($\mathbf{k}^I_{u,\hat{i}^t}$) as a justification at each turn. We also compare against a SOTA interactive recommender, **CE-VAE** [133], which learns a VAE with a bidirectional mapping between critique vectors and the user latent preference space. While we were unable to access to code for and replicate the results for the recent MM-VAE model [3], we note that for the Beer and Music domains (which we have in common) both of our bot-play models significantly out-perform MM-VAE's reported success rates for all N when all items can be recommended (Figure 4.4).

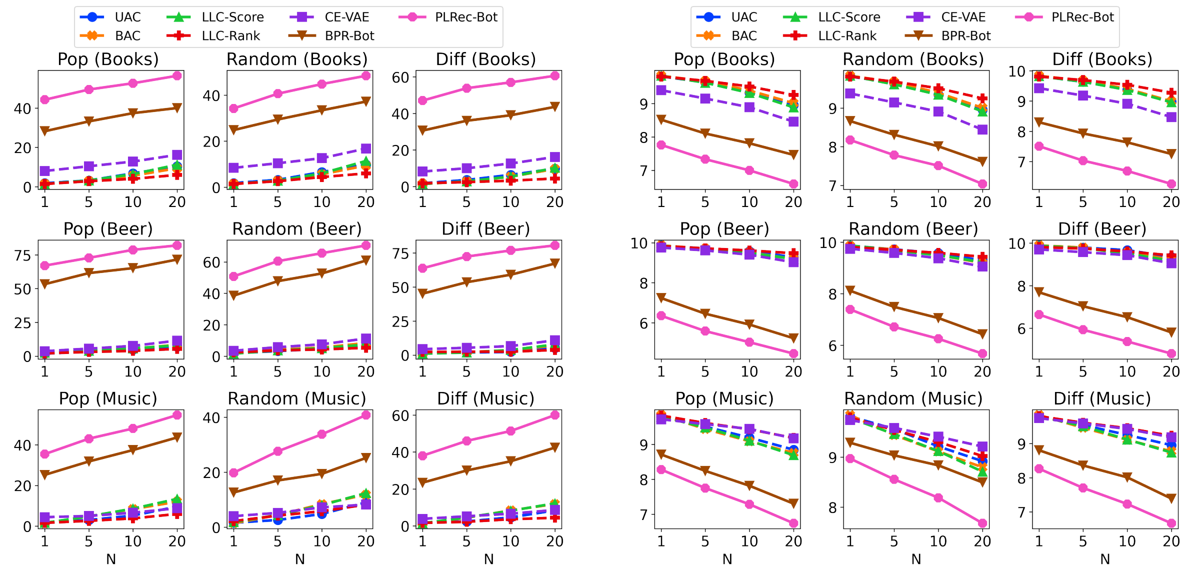
4.5 Experiments

4.5.1 RQ1: Can our framework enable multi-step critiquing?

Following standard practice [132, 110, 3], we measure multi-step critiquing performance via average success rate (SR@N)—the percentage of sessions where the target item reaches rank threshold N—and the average session length for the target to reach a rank threshold (Figure 4.4). We find that both of our candidate models (**BPR-Bot** and **PLRec-Bot**) out-perform all baselines. As our bot-play fine-tuning seeker model picks critiques by popularity, we expect our models to perform best in the Pop setting. However, BPR-Bot and PLRec-Bot succeed faster and at a higher rate than baselines in *all* user settings, including random critiquing with no prior on user behavior.

Linear critiquing models (UAC, BAC, LLC-Score/Rank) perform poorly on multi-step critiquing compared to models that can generate justifications, especially when trying to find the goal item outright (N=1). This suggests that personalized justifications help users choose more effective rationales to critique. Despite out-performing linear critiquing models, CE-VAE performs worse across all settings compared to models trained in our bot-play framework. This suggests that our models generate personalized justifications that are more helpful for narrowing down a user's preferences compared to CE-VAE. In Section 4.5.3, we further investigate the usefulness and accuracy of our rationales.

For **PLRec-Bot**, our base recommender system is initialized in the same way as the linear

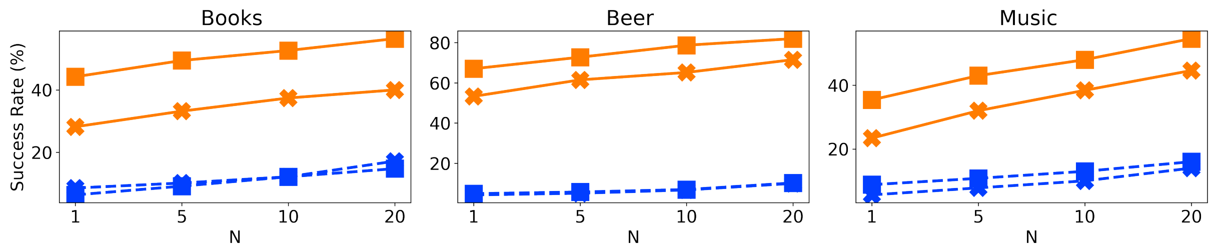


(a) Success Rate (%) @ N **(b)** Avg. # turns for target item to reach rank N

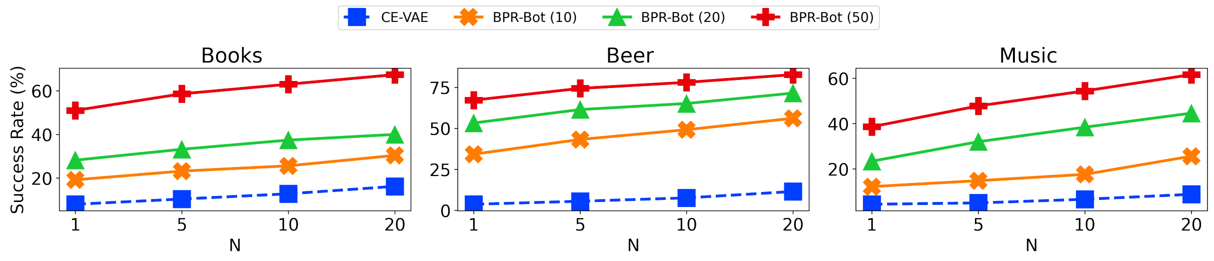
Figure 4.4. User simulation evaluation of our models—BPR-Bot (brown triangle) and PLRec-Bot (pink circle)—compared to linear critiquing and variational baselines for conversational recommendation (dashed lines). Models trained with our bot-play framework succeed at significantly higher rates (a) and surface desired items significantly faster (b) than all baselines.

critiquing models (UAC, BAC, LLC-Score/Rank). However, we observe an order of magnitude improvement in success rate across all rank thresholds N compared to linear models (and the similarly complex CE-VAE model). This demonstrates that we do not need to solve a linear programming problem for each critiquing step (like LLC-Score/Rank)—fine-tuning a model with our bot-play framework is more effective at teaching conversational agents to incorporate user feedback.

With **BPR-Bot**, we demonstrate that our bot-play framework can also be effectively applied to extremely lightweight and simple base recommender systems. Our base BPR models require an order of magnitude (5x-40x) fewer parameters than baseline models, representing users and items with only 20 latent dimensions. Nonetheless, by fine-tuning this model with our bot-play framework, we are able to again out-perform baselines by wide margins in all settings. Success with both PLRec-Bot and BPR-Bot showcases the model-agnostic nature of our framework, and in future work we hope to investigate its benefits with a wider range of base



(a) Bot-play models (orange) vs. non-bot-play ablations (blue)



(b) CE-VAE (dashed) vs. BPR-Bot for $h \in [10, 20, 50]$ (solid)

Figure 4.5. Success Rate @ N (% sessions where target item rank ≤ N) for ablation settings: (a) Bot-play improves target item ranking across datasets compared to the ablation for PLRec-Bot (squares) and BPR-Bot (crosses). (b) As latent dimension grows ($h \uparrow$), bot-play fine-tuning confers greater benefits. All models, including extremely lightweight $h = 10$ out-perform the best baseline model (CE-VAE).

recommender systems.

Overall, our models can better assist users with varying levels of domain knowledge and specific preferences compared to SOTA methods for conversational critiquing. We have thus shown that our bot-play framework enables the training of multi-turn conversational recommenders *without the need for costly supervised dialog transcripts*.

4.5.2 RQ2: Does *bot-play* help improve multi-step critiquing ability?

We next demonstrate that our bot-play fine-tuning is responsible for gains in multi-step critiquing performance (Figure 4.5a) by comparing BPR-Bot (crosses) and PLRec-Bot (squares) against ablated versions that were trained using the first step of our framework but *not* fine-tuned via bot-play. For clarity, we display only results using the Pop user behavioral model, as we observe the same trends with all three user models.

Bot-play confers a noticeable benefit for both BPR-Bot (100-300% improvement in success rate for various N) and PLRec-Bot (250-400% improvement) across domains, with the largest improvements observed with the Beer domain. This may be due to relatively dense occurrence of rationales in user reviews, with an average of 7.4 unique rationales expressed in each review (Table 4.3). This demonstrates that we can effectively train conversational recommender systems using our bot-play framework using domains with user reviews in lieu of crowd-sourced dialog transcripts.

In domains with more sparse coverage of subjective rationales (i.e. Books with 1.8 rationales/review and Music with 2.5 rationales/review), we observe lower improvement when using bot-play—our model may encounter insufficient cases of rare rationales being critiqued. This seems to affect lightweight models (BPR-Bot) much more than more complex base recommender systems (PLRec-Bot). In future work, we will explore adding noise to our user model to ensure that the bot-play process encounters more rare rationales.

We next investigate whether our framework is model size-agnostic. We fine-tune BPR models of varying sizes (varying user/item representation dimensionality h between 10 and 50), with success rates shown in 4.5b. We see that regardless of model size, simple recommender systems fine-tuned under our framework out-perform state-of-the-art conversational critiquing methods (CE-VAE). Models with higher latent dimensionality ($h = 10 \rightarrow 20 \rightarrow 50$) benefit more from bot-play, suggesting that our method learns to effectively navigate complex preference spaces.

The marginal benefit of increasing latent dimensionality seems to slow for the Beer domain (with the highest density of rationales per review, item, and user), while we continue to observe large benefits from increasing model size in Books and Music. This suggests that our bot-play framework allows large models to more effectively learn to encode user feedback in domains with sparse user feedback.

Finally, we consider conversational recommendation with multiple simultaneous critiques. As we observe in our user studies (Section 4.6), people tend to give multiple pieces of feedback at a given turn, with an average of around 2 critiques. As our bot-play training (Algorithm 2)

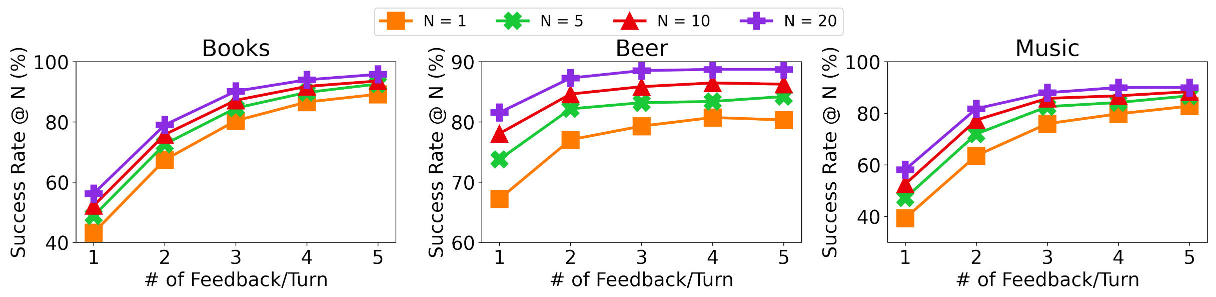


Figure 4.6. Success Rate @ N for PLRec-Bot with a maximum of 1 to 5 critiques at each turn. Despite our bot-play training using only a single critique per turn, each additional piece of feedback provides additional improvements in success rate, reaching 60%+ success rates at 1 for 2 pieces of feedback.

only simulates a single critique per turn, we investigate whether such a model can handle more realistic behavior.

In Figure 4.6, we plot the success rate at N for N$\in$ $[1, 5, 10, 20]$ for different numbers of critiques per turn. Our bot-play successfully allows our model to appropriately react to user behavior with varying degrees of feedback—the marginal value of each additional piece of feedback per turn is fairly high for the second and third pieces of feedback. Indeed, while the success rate at 1 (rate at which our agent returns the goal item exactly within the turn limit) varies between 40-68% across datasets, adding an additional piece of feedback improves this to 65-77%. Our models can quickly narrow down the most appropriate candidate items, approaching 90-100% success rate for N=20.

We thus confirm that our method is model-agnostic, as it improves recommendation success rates for both the matrix factorization-based (BPR) and linear (PLRec) recommender systems. Similarly, we have shown that our bot-play method is size-agnostic, and is generally applicable to base recommender systems with any latent dimensionality. Finally, we observe that our bot-play fine-tuning allows our model to accommodate multiple simultaneous critiques per turn—suggesting its usefulness in real-world scenarios.

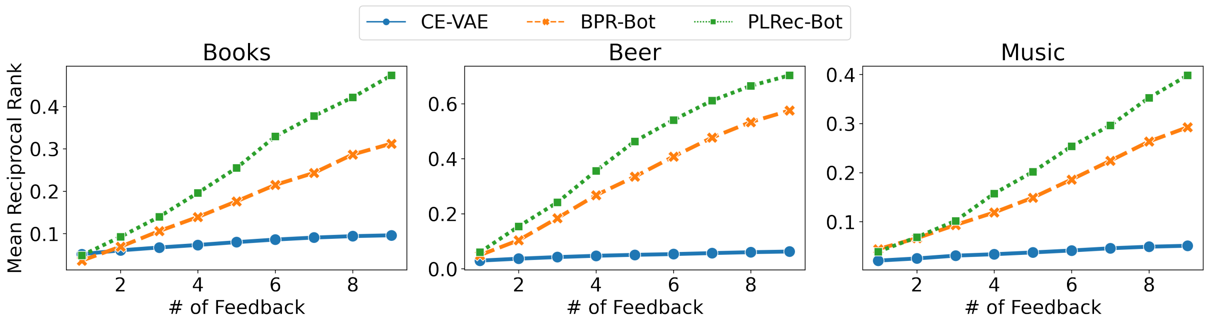


Figure 4.7. Mean Reciprocal Rank (MRR) vs. pieces of user feedback received, comparing the best baseline (CE-VAE, blue circles) against BPR-Bot (orange crosses) and PLRec-Bot (green squares). Users are able to give much more useful feedback when presented with rationales for both of our models, improving MRR faster than CE-VAE.

4.5.3 RQ3: Can our models generate useful and accurate rationales?

We next explore whether our model is surfacing appropriate rationales to guide the user and elicit feedback. We evaluate two main criteria with regards to rationales: 1) *usefulness*, or whether the rationales can help the user give effective feedback to more easily find their desired item; and 2) *accuracy*, or whether our model surfaces rationales related to the user's true preferences in that session.

We note that accuracy and usefulness of rationales must be balanced in a conversational critiquing system. This is because a user's reviews are necessarily incomplete: the user is unlikely to take the time to express every single one of their opinions about a product—including subtle preferences that may help them decide between very similar items. As a result, the system must both predict the rationales a user would express in their review of the target item *and* the qualities specific to a recommended item that help users distinguish between similar items.

To measure the **usefulness** of our rationales, we measure the mean reciprocal rank (MRR) of the target item for each piece of feedback given by the user. This reflects the value of each piece of feedback: we desire a model that can properly incorporate user feedback to more quickly identify the user's real preference (improve the goal item rank and MRR). In Figure 4.7, we plot the MRR against pieces of user feedback for PLRec-Bot (squares) and BPR-Bot (crosses)

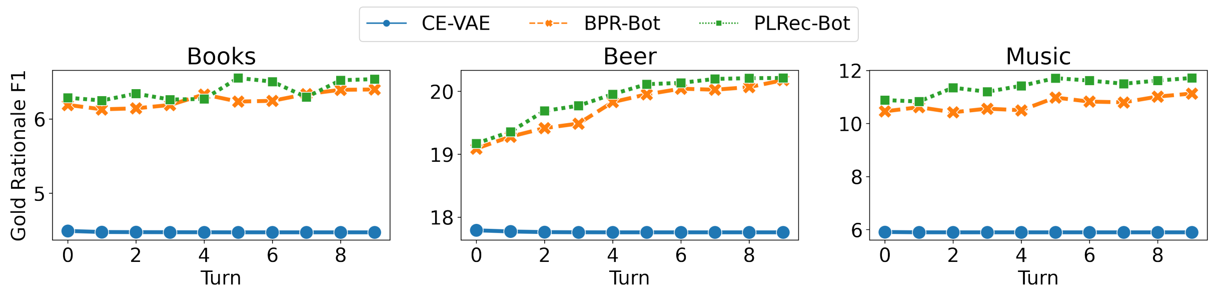


Figure 4.8. F1 score of rationales surfaced by conversational recommender systems compared to the user's ground truth rationales of the target item. When comparing CE-VAE (blue circles) to models trained with our bot-play framework—BPR-Bot (orange crosses) and PLRec-Bot (green squares)—our models more accurately infer the user's session preferences, and can improve their accuracy with each piece of user feedback.

compared to the best baseline conversational critiquing system (CE-VAE). We see that as the conversation progresses, models trained with our bot-play framework can more accurately rank the user's preferred items compared to CE-VAE.

More importantly, the "slope" of this graph represents the *marginal value* of each piece of feedback. For both PLRect-Bot and BPR-Bot, we observe a significantly higher marginal value of user feedback, suggesting that our rationales are more useful than those surfaced by CE-VAE. We also find that the marginal value of user feedback stays roughly constant for each piece of feedback, showing that our models can effectively refine user preferences even if a user has already provided several pieces of feedback.

We next measure the **accuracy** of rationales surfaced by conversational recommender systems. We assume that when writing a review, the user faithfully expresses their true preferences via the rationales contained in the review. As such, for each session where a user u tries to find item i, we take as ground truth the rationales extracted from the user's true review of the target item $\mathbf{k}_{u,i}$. In Figure 4.8, we plot the average F1 score of the rationales presented to the user (compared to the ground truth session preferences) at each turn of conversation for BPR-Bot, PLRec-Bot, and the CE-VAE baseline.

Across all datasets, we find that bot-play models provide more accurate justifications compared to CE-VAE. Furthermore, unlike CE-VAE, the accuracy of our justifications tends to increase as the session progresses. This suggests that when receiving feedback from the user, our models can improve their understanding of the user's preference in that particular session. This may help reinforce the user's trust of our system, as it provides the sense of an agent who "learns" the user's preferences during a conversation.

We note that models are able to better refine rationales in domains with more dense expression of subjective rationales per user review (Table 4.3). In particular, the book domain contains both the most users and the lowest density of rationales per review, and our models see the least improvement in F1 score over a conversation. On the other hand, this may reflect how our models suggest more rationales than users typically reveal, in order to help users better evaluate suggested novels.

4.6 Human Study

4.6.1 Human Evaluation

Following Li et al. [112], we conduct a comparative evaluation of 100 simulated user sessions on four criteria: which agent seems more useful, informative, knowledgeable and adaptive. We compare each bot-play model (**BPR-Bot** and **PLRec-Bot**) against an ablative version (with no bot-play) and the best baseline (CE-VAE).

Each sample is evaluated by three annotators, with all annotators recruited via the Amazon Mechanical Turk (MTurk) platform. We used crowd-workers with a historical 99% acceptance rate on their work to ensure quality, and crowd-workers were paid in excess of Federal minimum wage in the United States given the average time taken to complete an evaluation. We observe substantial [100] inter-annotator agreement, with Fleiss κ [51] of 0.67, 0.79, 0.73, and 0.60 for the usefulness, informativeness, knowledgeable, and adaptiveness criteria, respectively. Scores are shown in Table 4.4.

BPR-Bot and PLRec-Bot are judged to be significantly more informative and knowledgeable than ablative models and CE-VAE, showing that our models can accurately and convincingly

Table 4.4. Session-level human evaluation via ACUTE-EVAL. Users were asked which model was more Useful, Informative, Knowledgeable, and Adaptive when comparing bot-play models against CE-VAE and an ablative baseline with no bot-play fine-tuning. Results are shown for BPR-Bot (top) and PLRec-Bot (bottom). W/L percentages are reported while ties are not. All results statistically significant with $p < 0.05$.

BPR-Bot	Useful		Informative		Knowledgeable		Adaptive	
vs	W	L	W	L	W	L	W	L
Ablation (BPR)	**78***	10	**73***	11	**68***	15	**85***	5
CE-VAE	**83***	9	**74***	10	**63***	16	**81***	8
PLRec-Bot	Useful		Informative		Knowledgeable		Adaptive	
vs	W	L	W	L	W	L	W	L
Ablation (PLRec)	**86***	5	**78***	7	**74***	8	**81***	9
CE-VAE	**87***	7	**79***	11	**77***	12	**83***	10

explain each suggestion. This supports our findings from user simulations in Section 4.5.3. In particular, wins in informativeness and knowledgeability reflect how rationales surfaced by our models accurately describe the subjective opinions of users regarding the suggested item. If users believe a conversational agent can both accurately describe an item and reflect their personal opinions, they are more likely to trust the system and continue to interact with the agent in a meaningful way [204].

The usefulness and adaptiveness criteria capture how models help the user achieve their end goal (i.e. finding the most relevant item in as few turns as possible). Bot-play models are judged to be more useful than alternatives and follow critiques more consistently when adapting recommendations. This again suggests that users 1) trust our models' rationales for recommendations and 2) can meaningfully interact with our model to achieve their end goal.

Our framework allows us to train conversational agents that are useful and engaging for human users: evaluators overwhelmingly judged the models trained via bot-play to be more useful, informative, knowledgeable, and adaptive compared to CE-VAE and ablated variants.

Table 4.5. Cold-start user study results. On a per-turn basis, users found our bot-play model to be significantly ($p < 0.01$) more useful, informative, and adaptive compared to the baseline. On a session basis, significantly more users ($p < 0.01$) would use the bot-play model "often" or "always" to receive book recommendations compared to the baseline.

	Avg. Feedback	Useful	Informative	Adaptive	Use Again
Ablation (No Bot Play)	1.77 ± 0.08	0.67 ± 0.24	0.75 ± 0.21	0.64 ± 0.27	41%
Our Method	2.05 ± 0.13	**0.79 ± 0.24***	**0.88 ± 0.18**	**0.78 ± 0.23***	**69%***

4.6.2 Cold-Start User Study

We conduct a user study using the Books dataset to evaluate if our model is a useful real-time conversational recommender. In particular, we wish to see if models trained with bot-play using user reviews could effectively make use of feedback from new users (cold-start). We recruited 64 native English speakers from universities across the United States, randomly assigning half to interact with **BPR-Bot** and half to interact with the ablation (no bot-play).

We initialize each session with the mean of all learned user embeddings to provide the same initial set of suggestions for each new user. At each turn, the user sees the three top-ranked items with justifications (rationales) and can critique multiple rationales. On average, users critiqued two rationales per turn—this suggests that when training interactive agents we can assume multiple critiques at each turn. In future work, we aim to study whether users in warm-start and cold-start situations give differing amounts of feedback at each turn of conversation.

At each turn, we again follow Li et al. [112] to ask users if the generated explanations are *informative*, *useful* in helping to make a decision, and whether our system correctly *adapted* its suggestions in response to the user's feedback. We provide four options for each question: no, weak-no, weak-yes, and yes. We then map these values to a score between 0 and 1 [87], with normalized scores for each question shown in Table 4.5. **BPR-Bot** significantly out-scores the ablation in all three metrics ($p < 0.01$), showing that fine-tuning via our bot-play framework instills a stronger ability to respond to critiques and provide meaningful explanations—even for new users.

At the end of a session, we additionally ask the user how frequently (if at all) they would choose to engage with our interactive agent in their daily life. Users preferred BPR-Bot by significant margins—69% indicated they would “often” or “always” use BPR-Bot to find books compared to 41% for the ablation. We are encouraged that over two thirds of users would regularly use our system, and it confirms that our critiquing approach to conversational recommendation reflects a realistic and appealing human interaction paradigm.

4.7 Conclusion

In this chapter we develop conversational recommenders that can engage with users over multiple turns, providing rationales for suggestions and incorporating user feedback. We present a model-agnostic framework to train conversational agents in this modality via self-supervised bot-play in any domain using only review data. We use two popular underlying recommender systems to train the **BPR-Bot** and **PLRec-Bot** agents using our framework, showing quantitatively on three datasets that our models 1) offer superior multi-turn recommendation performance compared to current SOTA methods; 2) provide more useful and informative rationales for each recommended item compared to current SOTA methods; and 3) can effectively refine suggestions in real-time, as shown in user studies. We further show that our bot-play framework confers its benefits for models with different underlying architectures and levels of complexity. In future work, we aim to adapt our framework to free-form natural language critiques, allowing users to more flexibly express feedback.

4.8 Acknowledgements

This chapter contains material as it appears in and adapted from the Second Workshop on Trustworthy Natural Language Processing at NAACL 2022 (“Self-Supervised Bot Play for Transcript-Free Conversational Recommendation with Rationales” Shuyang Li, Bodhisattwa Prasad Majumder, Julian McAuley). The dissertation author was the primary investigator and

author of this paper.

Part II

Personalized Interactive Agents for Instructional Texts

Chapter 5

Personalized Cooking Recipe Generation from Historical Interactions

5.1 Introduction

In the kitchen, we increasingly rely on instructions from cooking websites: recipes. A cook with a predilection for Asian cuisine may wish to prepare chicken curry, but may not know all necessary ingredients apart from a few basics. These users with limited knowledge cannot rely on existing recipe generation approaches that focus on creating coherent recipes given all ingredients and a recipe name [92]. Such models do not address issues of personal preference (e.g. culinary tastes, garnish choices) and incomplete recipe details. **In this chapter,** we propose to approach both problems via *personalized generation* of plausible, user-specific recipes using user preferences extracted from previously consumed recipes.

Our work combines two important tasks from natural language processing and recommender systems: data-to-text generation [56] and personalized recommendation [173]. Our model takes as user input the name of a specific dish, a few key ingredients, and a calorie level. We pass these loose input specifications to an encoder-decoder framework and attend on user profiles—learned latent representations of recipes previously consumed by a user—to generate a recipe *personalized* to the user's tastes. We fuse these 'user-aware' representations with decoder output in an attention fusion layer to jointly determine text generation. Quantitative (perplexity, user-ranking) and qualitative analysis on user-aware model outputs confirm that personalization indeed assists in generating plausible recipes from incomplete ingredients.

While personalized text generation has seen success in conveying user writing styles in the product review [150, 151] and dialogue [243] spaces, we are the first to consider it for the problem of recipe generation, where output quality is heavily dependent on the *content* of the instructions—such as ingredients and cooking techniques.

To summarize, our main contributions are as follows:

1. We explore a new task of generating plausible and personalized recipes from incomplete input specifications by leveraging historical user preferences;

2. We release a new dataset of 180K+ recipes and 700K+ user reviews for this task;[1]

3. We introduce new evaluation strategies for generation quality in instructional texts, centering on quantitative measures of coherence. We also show qualitatively and quantitatively that personalized models generate high-quality and specific recipes that align with historical user preferences.

5.2 Related Work

Large-scale transformer-based language models have shown surprising expressivity and fluency in creative and conditional long-text generation [210, 169]. Recent works have proposed hierarchical methods that condition on narrative frameworks to generate internally consistent long texts [46, 229, 238]. Here, we generate procedurally structured recipes instead of free-form narratives.

Recipe generation belongs to the field of data-to-text natural language generation [56], which sees other applications in automated journalism [107], question-answering [2], and abstractive summarization [158], among others. Kiddon et al. [91], Bosselut et al. [15] model recipes as a structured collection of ingredient entities acted upon by cooking actions. Kiddon et al. [92] imposes a 'checklist' attention constraint emphasizing hitherto unused ingredients

[1]https://www.kaggle.com/datasets/shuyangli94/food-com-recipes-and-user-interactions

during generation. Yang et al. [237] attend over explicit ingredient references in the prior recipe step. Similar hierarchical approaches that infer a full ingredient list to constrain generation will not help personalize recipes, and would be infeasible in our setting due to the potentially unconstrained number of ingredients (from a space of 10K+) in a recipe. We instead learn historical preferences to guide full recipe generation.

A recent line of work has explored user- and item-dependent aspect-aware review generation [151, 150]. This work is related to ours in that it combines contextual language generation with personalization. Here, we attend over historical user preferences from previously consumed recipes to generate recipe content, rather than writing styles.

5.3 Approach

Our model's input specification consists of: the recipe name as a sequence of tokens, a partial list of ingredients, and a caloric level (high, medium, low). It outputs the recipe instructions as a token sequence: $\mathcal{W}_r = \{w_{r,0}, \ldots, w_{r,T}\}$ for a recipe r of length T. To personalize output, we use historical recipe interactions of a user $u \in \mathcal{U}$.

5.3.1 Encoder

Our encoder has three embedding layers: vocabulary embedding $\mathcal{V}$, ingredient embedding $\mathcal{I}$, and caloric-level embedding $\mathcal{C}$. Each token in the (length L_n) recipe name is embedded via $\mathcal{V}$; the embedded token sequence is passed to a two-layered bidirectional GRU (BiGRU) [28], which outputs hidden states for names $\{\mathbf{n}_{\text{enc},j} \in \mathbb{R}^{2d_h}\}$, with hidden size d_h. Similarly each of the L_i input ingredients is embedded via $\mathcal{I}$, and the embedded ingredient sequence is passed to another two-layered BiGRU to output ingredient hidden states as $\{\mathbf{i}_{\text{enc},j} \in \mathbb{R}^{2d_h}\}$. The caloric level is embedded via $\mathcal{C}$ and passed through a projection layer with weights W_c to generate calorie hidden representation $\mathbf{c}_{\text{enc}} \in \mathbb{R}^{2d_h}$.

5.3.2 Ingredient Attention

We apply attention [7] over the encoded ingredients to use encoder outputs at each decoding time step. We define an attention-score function α with key K and query Q:

$$\alpha(K, Q) = \frac{\exp\left(\tanh\left(W_\alpha\left[K + Q\right] + \mathbf{b}_\alpha\right)\right)}{Z},$$

with trainable weights W_α, bias $\mathbf{b}_\alpha$, and normalization term Z. At decoding time t, we calculate the ingredient context $\mathbf{a}_t^i \in \mathbb{R}^{d_h}$ as:

$$\mathbf{a}_t^i = \sum_{j=1}^{L_i} \alpha\left(\mathbf{i}_{\text{enc},j}, \mathbf{h}_t\right) \times \mathbf{i}_{\text{enc},j}.$$

5.3.3 Decoder

The decoder is a two-layer GRU with hidden state h_t conditioned on previous hidden state h_{t-1} and input token $w_{r,t}$ from the original recipe text. We project the concatenated encoder outputs as the initial decoder hidden state:

$$\begin{aligned}
\mathbf{h}_0 \left(\in \mathbb{R}^{d_h}\right) &= W_{h_0}\left[\mathbf{n}_{\text{enc},L_n}; \mathbf{i}_{\text{enc},L_i}; \mathbf{c}_{\text{enc}}\right] + \mathbf{b}_{h_0} \\
\mathbf{h}_t, \mathbf{o}_t &= \text{GRU}\left(\left[w_{r,t}; \mathbf{a}_t^i\right], \mathbf{h}_{t-1}\right).
\end{aligned}$$

To bias generation toward user preferences, we attend over a user's previously reviewed recipes to jointly determine the final output token distribution. We consider two different schemes to model preferences from user histories: (1) recipe interactions, and (2) techniques seen therein (defined in Section 5.4). Rendle et al. [175], Quadrana et al. [165], Ueda et al. [207] explore similar schemes for personalized recommendation.

5.3.4 Prior Recipe Attention

We obtain the set of prior recipes for a user u: R_u^+, where each recipe can be represented by an embedding from a recipe embedding layer $\mathcal{R}$ or an average of the name tokens embedded by $\mathcal{V}$. We attend over the k-most recent prior recipes, R_u^{k+}, to account for temporal drift of user preferences [147]. These embeddings are used in the '**Prior Recipe/Name**' models, respectively.

Given a recipe representation $\mathbf{r} \in \mathbb{R}^{d_r}$ (where d_r is recipe- or vocabulary-embedding size depending on the recipe representation) the *prior recipe attention* context $\mathbf{a}_t^{r_u}$ is calculated as

$$\mathbf{a}_t^{r_u} = \sum_{r \in R_u^{k+}} \alpha\left(\mathbf{r}, \mathbf{h}_t\right) \times \mathbf{r}.$$

5.3.5 Prior Technique Attention

We calculate prior technique preference (used in the '**Prior Tech**' model) by normalizing co-occurrence between users and techniques seen in R_u^+, to obtain a preference vector $\boldsymbol{\rho}_u$. Each technique x is embedded via a technique embedding layer $\mathcal{X}$ to $\mathbf{x} \in \mathbb{R}^{d_x}$. *Prior technique attention* is calculated as

$$\mathbf{a}_t^{x_u} = \sum_{x \text{ seen in } R_u^+} \left(\alpha\left(\mathbf{x}, \mathbf{h}_t\right) + \rho_{u,x}\right) \times \mathbf{x},$$

where, inspired by copy mechanisms [183, 64], we add $\rho_{u,x}$ for technique x to emphasize the attention by the user's prior technique preference.

5.3.6 Attention Fusion Layer

We fuse all contexts calculated at time t, concatenating them with decoder GRU output and previous token embedding:

$$\mathbf{a}_t^f = \mathrm{ReLU}\left(W_f\left[w_{r,t}; \mathbf{o}_t; \mathbf{a}_t^i; (\mathbf{a}_t^{r_u} \text{ or } \mathbf{a}_t^{x_u})\right] + \mathbf{b}_f\right).$$

Table 5.1. Food.com data statistics for the (a) raw/processed corpus and (b) train/dev/test splits

(a) Raw and Processed Statistics

	Recipes	Users	Reviews
Raw	231,637	226,570	1,132,367
Processed	178,265	25,076	749,053

(b) Data Splits

Split	Users	Recipes	Actions	Sparsity
Train	25,076	160,901	698,901	99.983%
Dev	7,023	6,621	7,023	–
Test	12,455	11,695	12,455	–

We then calculate the token probability:

$$P(S_{r,t}) = \text{softmax}\left(W_P[\mathbf{a}_t^f] + \mathbf{b}_P\right),$$

and maximize the log-likelihood of the generated sequence conditioned on input specifications and user preferences. Figure 5.1 shows a case where the Prior Name model attends strongly on previously consumed savory recipes to suggest the usage of an additional ingredient ('cilantro').

5.4 Recipe Dataset: Food.com

We collect a novel dataset of 230K+ recipe texts and 1M+ user interactions (reviews) over 18 years (2000-2018) from Food.com.[2] Here, we restrict to recipes with at least 3 steps, and at least 4 and no more than 20 ingredients. We discard users with fewer than 4 reviews, giving 180K+ recipes and 700K+ reviews, with splits as in Table 5.1.

Our model must learn to generate from a diverse recipe space: in our training data, the average recipe length is 117 tokens with a maximum of 256. There are 13K unique ingredients across all recipes. Rare words dominate the vocabulary: 95% of words appear <100 times, accounting for only 1.65% of all word usage. As such, we perform Byte-Pair Encoding (BPE) tokenization [185, 167], giving a training vocabulary of 15K tokens across 19M total mentions. User profiles are similarly diverse: 50% of users have consumed ≤6 recipes, while 10% of users have consumed >45 recipes.

[2]https://www.kaggle.com/shuyangli94/food-com-recipes-and-user-interactions

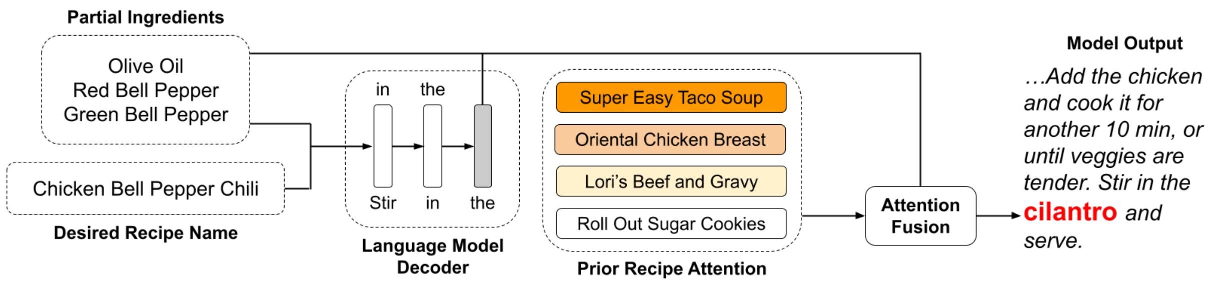


Figure 5.1. Sample data flow through model architecture. User has bell peppers and olive oil, and wants to make Chicken Bell Pepper Chili. The model attends over the user's previously reviewed recipes, focusing strongest (darkest color) on the Mexican and Asian recipes. As a result, the model recommends to supplement the recipe with **cilantro**—a recipe used frequently in both tacos and Asian cuisine.

We order reviews by timestamp, keeping the most recent review for each user as the test set, the second most recent for validation, and the remainder for training (sequential leave-one-out evaluation [84]). We evaluate only on recipes not in the training set.

We manually construct a list of 58 cooking techniques from 384 cooking actions collected by Bosselut et al. [15]; the most common techniques (*bake*, *combine*, *pour*, *boil*) account for 36.5% of technique mentions. We approximate technique adherence via string match between the recipe text and technique list.

5.5 Experiments and Results

For training and evaluation, we provide our model with the first 3-5 ingredients listed in each recipe. We decode recipe text via top-k sampling [169], finding $k = 3$ to produce satisfactory results. We use a hidden size $d_h = 256$ for both the encoder and decoder. Embedding dimensions for vocabulary, ingredient, recipe, techniques, and caloric level are 300, 10, 50, 50, and 5 (respectively). For prior recipe attention, we set $k = 20$, the 80th %-ile for the number of user interactions. We use the Adam optimizer [96] with a learning rate of 10^{-3}, annealed with a decay rate of 0.9 [78]. We also use teacher-forcing [222] in all training epochs.

In this chapter, we investigate how leveraging historical user preferences can improve generation quality over strong baselines in our setting. We compare our personalized models

Table 5.2. Metrics on generated recipes from test set. D-1/2 = Distinct-1/2, UMA = User Matching Accuracy, MRR = Mean Reciprocal Rank, PP = Pairwise preference over baseline (evaluated for 310 recipe pairs per model).

Model	PPL	BLEU-1	BLEU-4	ROUGE-L	D-1	D-2	UMA	MRR	PP (%)
NN	–	20.279	0.465	16.871	0.931	9.394	0.100	0.293	–
Enc-Dec	9.611	28.391	**3.385**	**25.001**	0.220	1.928	0.100	0.293	–
Prior Tech	9.572	**28.864**	3.312	24.920	0.233	**2.158**	0.128	0.319	62.821
Prior Recipe	9.551	27.858	3.215	24.822	0.231	2.062	0.302	0.412	**66.026**
Prior Name	**9.516**	28.046	3.211	24.794	**0.233**	2.080	**0.505**	**0.628**	61.165

against two baselines. The first is a name-based Nearest-Neighbor model (**NN**). We initially adapted the Neural Checklist Model of Kiddon et al. [92] as a baseline; however, we ultimately use a simple Encoder-Decoder baseline with ingredient attention (**Enc-Dec**), which provides comparable performance and lower complexity. All personalized models outperform baseline in BPE perplexity (Table 5.2) with Prior Name performing the best. While our models exhibit comparable performance to baseline in BLEU-1/4 and ROUGE-L, we generate more diverse (Distinct-1/2: percentage of distinct unigrams and bigrams) and acceptable recipes. BLEU and ROUGE are not the most appropriate metrics for generation quality. A 'correct' recipe can be written in many ways with the same main entities (ingredients). As BLEU-1/4 capture structural information via n-gram matching, they are not correlated with subjective recipe quality. This mirrors observations from Baheti et al. [8], Fan et al. [46].

We observe that personalized models make more diverse recipes than baseline. They thus perform better in BLEU-1 with more key entities (ingredient mentions) present, but worse in BLEU-4, as these recipes are written in a personalized way and deviate from gold on the phrasal level. Similarly, the 'Prior Name' model generates more unigram-diverse recipes than other personalized models and obtains a correspondingly lower BLEU-1 score.

Table 5.3. Sample generated recipe. Emphasis on personalization and explicit ingredient mentions via highlights.

Input	**Name:** Pomberrytini; **Ingredients:** pomegranate-blueberry juice, cranberry juice, vodka ; **Calorie:** Low
Gold	Place everything except the orange slices in a cocktail shaker. Shake until well mixed and well chilled. Pour into martini glasses and float an orange slice in each glass.
Enc-Dec	Combine all ingredients. Cover and refrigerate. Serve with whipped topping.
Prior Tech	Combine all ingredients. Store in refrigerator. Serve over ice. Enjoy!
Prior Recipe	Pour the ice into a cocktail shaker. Pour in the vodka and vodka. Add a little water and shake to mix. Pour into the glass and garnish with a slice of orange slices. Enjoy!
Prior Name	Combine all ingredients except for the ice in a blender or food processor. Process to make a smooth paste and then add the remaining vodka and blend until smooth. Pour into a chilled glass and garnish with a little lemon and fresh mint.

5.5.1 Qualitative Analysis

We present sample outputs for a cocktail recipe in Table 5.3. Generation quality progressively improves from generic baseline output to a blended cocktail produced by our best performing model. Models attending over prior recipes explicitly reference ingredients. The Prior Name model further suggests the addition of lemon and mint, which are reasonably associated with previously consumed recipes like coconut mousse and pork skewers. We present another set of generated examples for waffles in Table 5.4.

5.5.2 Personalization

To measure personalization, we evaluate how closely the generated text corresponds to a particular user profile. We compute the likelihood of generated recipes using identical input specifications but conditioned on ten different user profiles—one 'gold' user who consumed the original recipe, and nine randomly generated user profiles. Following Fan et al. [46], we expect the highest likelihood for the recipe conditioned on the gold user. We measure user matching accuracy (UMA)—the proportion where the gold user is ranked highest—and Mean Reciprocal Rank (MRR) [166] of the gold user. All personalized models beat baselines in both metrics,

Table 5.4. Sample generated waffle recipe for all models.

Input	**Name:** Cinna Nilla Waffles **Ingredients:** flour, egg, milk, vegetable oil, sugar **Calorie:** Medium
Gold	Heat waffle iron. Beat eggs in large bowl with hand beater until fluffy. Beat in remaining ingredients until smooth. Pour batter into waffle iron and cook for approximately 5 minutes. Serve immediately or cool to pack into freezer bags and freeze.
Enc-Dec	Mix the flour, baking soda, and salt in a bowl. In a separate bowl, whisk together the milk, oil, eggs and sugar. Stir the flour mixture into the egg mixture, and continue to mix well. Add the flour to the egg mixture. Mix well and pour into a greased waffle iron. Cook for 2 minutes, remove from heat and serve.
Prior Tech	In a medium bowl mix flour, eggs, and milk until combined. Add the dry ingredients and stir until just combined and do not mix. Heat griddle over medium heat, add the oil, oil, and cook the pancakes until golden brown and cooked through. Serve with a little milk or cream. Enjoy
Prior Recipe	In a mixing bowl, whisk together the eggs, milk, oil, sugar, vanilla, salt and vanilla. Cover and let the mixture stand in the fridge for about 1 hour. Spoon batter into waffle iron and close the grill.
Prior Name	Preheat waffle iron. Beat together the eggs, milk and oil until well blended, add the vanilla and mix well with a mixer. Fold in flour, baking powder, and cinnamon. Spread 1 / 2 the mixture in a greased waffle iron. Bake until golden brown, about 15 minutes per side. Sprinkle with powdered sugar and serve warm.

showing our models personalize generated recipes to the given user profiles. The Prior Name model achieves the best UMA and MRR by a large margin, revealing that prior recipe names are strong signals for personalization. Moreover, the addition of attention mechanisms to capture these signals improves language modeling performance over a strong non-personalized baseline.

Table 5.5. Coherence metrics on generated recipes from test set. Applying personalization via attention over any form of the user's interaction history improves recipe coherence, with recipe names having the most impact.

	Enc-Dec	Prior Tech	Prior Recipe	**Prior Name**
Recipe-Level Coherence	1.77	1.78	1.80	**1.82**
Recipe Step Entailment	0.72	0.73	0.76	**0.78**

5.5.3 Recipe Level Coherence

A plausible recipe should possess a coherent step order, and we evaluate this via a metric for recipe-level coherence. We use the neural scoring model from Bosselut et al. [14] to measure recipe-level coherence for each generated recipe. Each recipe step is encoded by BERT [38]. Our scoring model is a GRU network that learns the overall recipe step ordering structure by minimizing the cosine similarity of recipe step hidden representations presented in the correct and reverse orders. Once pretrained, our scorer calculates the similarity of a generated recipe to the forward and backwards ordering of its corresponding gold label, giving a score equal to the difference between the former and latter. A higher score indicates better step ordering (with a maximum score of 2). Table 5.5 shows that our personalized models achieve average recipe-level coherence scores of 1.78-1.82, surpassing the baseline at 1.77.

5.5.4 Recipe Step Entailment

Local coherence is also crucial to a user following a recipe: it is crucial that subsequent steps are logically consistent with prior ones. We model local coherence as an entailment task: predicting the likelihood that a recipe step follows the preceding. We sample several consecutive (positive) and non-consecutive (negative) pairs of steps from each recipe. We train a BERT [38] model to predict the entailment score of a pair of steps separated by a `[SEP]` token, using the final representation of the `[CLS]` token. The step entailment score is computed as the average of scores for each set of consecutive steps in each recipe, averaged over every generated recipe for a model, as shown in Table 5.5.

5.5.5 Human Evaluation

We presented 310 pairs of recipes for pairwise comparison [46] between baseline and each personalized model, with results shown in Table 5.2. On average, human evaluators preferred personalized model outputs to baseline 63% of the time, confirming that personalized attention improves the semantic plausibility of generated recipes. We also performed a small-scale human coherence survey over 90 recipes, in which 60% of users found recipes generated by personalized models to be more coherent and preferable to those generated by baseline models.

5.6 Conclusion

In this chapter, we propose a novel task: to generate personalized recipes from incomplete input specifications and user histories. On a large novel dataset of 180K recipes and 700K reviews, we show that our personalized generative models can generate plausible, personalized, and coherent recipes preferred by human evaluators for consumption. We also introduce a set of automatic coherence measures for instructional texts as well as personalization metrics to support our claims. Our future work includes generating structured representations of recipes to handle ingredient properties and accounting for references to collections of ingredients (e.g. “dry mix").

5.7 Acknowledgements

This chapter contains material as it appears in and adapted from Empirical Methods in Natural Language Processing, 2019 (“Generating Personalized Recipes from Historical User Preferences” Bodhisattwa Prasad Majumder*, Shuyang Li*, Jianmo Ni, Julian McAuley). The dissertation author was the co-primary investigator and author of this paper.

Chapter 6

Assistive Recipe Editing

6.1 Introduction

In the previous chapter, we presented a system to generate novel recipes tailored for a user's historical preferences. These users (or their prospective diners) may also have additional dietary constraints governing what types of foods they can or want to eat. **In this chapter,** we aim to build a system that can *tailor* recipes to fit users' dietary preferences and restrictions.

Cooking has played an integral role in human civilization and evolution for over 1.8 million years [224]. A growing population follows some form of dietary restriction [59], with motivations ranging from socioeconomic to medical [189, 37]. Home cooks browsing recipes on online recipe aggregators may often encounter an interesting recipe that does not fit their dietary needs (e.g. vegetarianism), and would benefit from a way to *edit* that recipe to fit their needs. These restrictions are easy for users to specify, but existing recipe websites offer few options for users with dietary constraints—even those with common restrictions like gluten intolerance (Table 6.1). We see an opportunity to build a model that can adapt recipes into more appropriate forms given users' dietary constraints.

To help such users, we introduce the task of *controllable recipe editing*: editing a base recipe so that it satisfies a user-specified dietary constraint (Figure 6.1). Controllable recipe editing can help home cooks of any experience level find diverse ways to satisfy their dietary needs and help develop interactive, personalized products like meal kits to accommodate individual

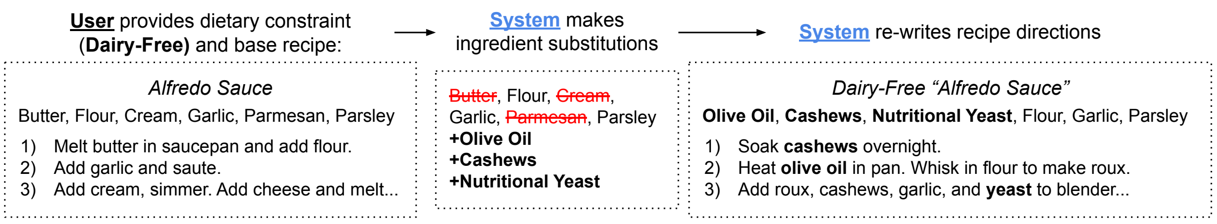

User provides dietary constraint (Dairy-Free) and base recipe:
System makes ingredient substitutions
System re-writes recipe directions
Alfredo Sauce
Butter, Flour, Cream, Garlic, Parmesan, Parsley
1) Melt butter in saucepan and add flour.
2) Add garlic and saute.
3) Add cream, simmer. Add cheese and melt...
~~Butter~~, Flour, ~~Cream~~, Garlic, ~~Parmesan~~, Parsley
+Olive Oil
+Cashews
+Nutritional Yeast
Dairy-Free "Alfredo Sauce"
Olive Oil, Cashews, Nutritional Yeast, Flour, Garlic, Parsley
1) Soak cashews overnight.
2) Heat olive oil in pan. Whisk in flour to make roux.
3) Add roux, cashews, garlic, and yeast to blender...

Figure 6.1. We investigate the task of controllable recipe editing: edit a base recipe to satisfy a given dietary restriction.

needs and preferences. This is a challenging task: rule-based substitution methods and existing recipe generators cannot adequately account for both the dietary and structural impacts of ingredient substitutions.

To tackle this task, we propose a **S**ystem for **H**ierarchical **A**ssistive **R**ecipe **E**diting (SHARE). We first use a Transformer [210] encoder-decoder to perform multiple simultaneous ingredient substitutions conditioned on a dietary restriction. We next employ a Transformer language model to write new instructions conditioned on the edited ingredients, using a copy mechanism [64] to increase ingredient-step coherence.

We conduct experiments on a novel dataset of 83K recipe pairs from user-reviewed recipes on a popular recipe site. Each pair is associated with a *dietary restriction* and contains a *base* recipe and a similar *target* recipe that satisfies the constraint. We evaluate edited recipes via automatic metrics to demonstrate that SHARE produces diverse and high-fidelity recipes. We survey 672 home cooks to assess the quality of our edited recipes compared to human-written recipes, finding that SHARE consistently produced the highest quality edits—in the process discovering several stylistic and structural aspects that cooks evaluate when searching for recipes. We also recruit seven home cooks to cook 21 recipes generated from SHARE and evaluate both the cooking process and final product, showing that such recipes are delicious, easy to make, and satisfy dietary restrictions.

We summarize our main contributions in this chapter: 1) We propose the *controllable recipe editing* task to assist an under-served population of home cooks by editing recipes to satisfy their dietary constraints; 2) We create the novel **RecipePairs** dataset—containing 83K pairs of recipes and versions thereof satisfying a dietary constraint—to facilitate recipe editing; 3) We train a hierarchical model for controllable recipe editing (SHARE), finding via quantitative trials, human studies, and real-world cooking trials that SHARE creates more coherent and constraint-respecting recipes compared to strong baselines—and that home cooks find its dishes appealing.

6.2 Related Work

We specifically aim to help a population under-served by existing resources: people with dietary restrictions. Recent works in nutritional recipe recommendation [26, 61] aim to recommend generally healthier food options. To our knowledge, while prior work has focused on ingredient substitution *or* generating recipe directions, ours is the first to examine both to create a complete recipe. We are motivated by work on single ingredient substitution [233, 200], but our system accommodates multiple simultaneous substitutions when predicting a target ingredient set. Recent work in recipe generation has focused on improving coherence [15] and ingredient specificity [92]. We draw from these as well as sentence editing work that uses retrieved *prototype sequences* to control generated text [65, 68], to improve conditioning on ingredients.

The editing of procedural texts like recipes resembles conditional text generation and paraphrase generation. Transformer [210] language models have seen success in both fields [89, 223], and Raffel et al. [170] demonstrated that such models can make use of a set of control codes for simultaneous multi-task learning. Lee et al. [102] train a single model for two tasks—ingredient extraction and recipe step generation conditioned on the recipe directions and ingredients, respectively. A user must thus provide either the complete set of ingredients or a full set of directions—neither of which may be known to home cooks looking for ways to adapt a recipe to their individual needs. Majumder et al. [136] explore ways to use browsing histories to personalize recipes for cooking website users. Their model uses these histories, a recipe name, and ingredients as input to generate personalized instructions, but cannot be controlled to

Table 6.1. For each soft (low-carb, low-calorie, low-fat, low-sugar) and hard (vegetarian, gluten-free, dairy-free) dietary constraint: users searching for such recipes, number of recipes satisfying constraint, pairs in RecipePairs, ingredient substitution rules, and prohibited ingredients.

	Low-Carb	Low-Cal.	Low-Fat	Low-Sug.	Vegetarian	Gluten-Free	Dairy-Free
% Users	71.3	63.8	51.5	21.9	59.9	24.1	19.7
% Recipes	16.9	14.3	8.6	2.6	13.3	2.3	1.5
Pairs	19 K	17 K	13 K	6 K	18 K	6 K	4 K
Rules	10	78	55	11	83	18	14
Banned	–	–	–	–	252	137	112

generate constraint-satisfying (e.g. gluten-free) recipes.

6.3 RecipePairs Dataset

Several recipe corpora have been collected, including the 150K-recipe *Now You're Cooking!* dataset [92, 15], Recipe1M+ [140] for cross-modal retrieval tasks, and the Food.com [136] dataset. We extend the Food.com dataset, aggregating user-provided category tags for each of the 459K recipes—comprising soft constraints (low-sugar, low-fat, low-carb, low-calorie) and strict/hard dietary restrictions (gluten-free, dairy-free, vegetarian). We compile a list of 501 ingredients that violate hard constraints following medical website guidance,[1] further filtering out any tagged recipes that contain banned ingredients to ensure each of our category tags is accurate.

We pair recipes into a *base* recipe that does not satisfy any *category* (dietary constraint) and a *target* version that satisfies the constraint. To ensure base and target recipes are similar, we pair recipes by name, sampling target recipes whose name contains a base recipe name (e.g. 'healthy oat chocolate cookie' as a gluten-free version of 'chocolate cookie'), resulting in 83K total pairs in RecipePairs comprising 36K unique base recipes and 60K unique targets. We split the data into 81K pairs for training and 1K disjoint pairs each for testing and validation. The large ingredient space contributes to the difficulty of the task. We normalize ingredients by

[1]e.g. https://www.ncbi.nlm.nih.gov/books/NBK310258/

stripping brand names and amounts (e.g. ‘Knoxville Farms’ or ‘1 tablespoon’) and remove rare variants, resulting in 2,942 unique ingredients that appear across 10+ recipes in our dataset. In Table 6.1, we show the number of recipe pairs targeting each dietary constraint and the number of banned ingredients for each hard constraint.

6.3.1 Recipes Satisfying Dietary Restrictions

To better understand the audience that would benefit from a recipe editing application, we first investigate how well users of Food.com are served when searching for recipes satisfying dietary constraints. A long tail of dietary constraints are poorly supported by the site. While 93% of users look for recipes that satisfy some dietary constraint, only 43% of all recipes satisfy a dietary restriction—the stark difference between the proportion of users looking to follow a dietary constraint and the proportion of recipes accommodating each constraint is seen in Table 6.1. In particular, low-sugar, dairy-free, and gluten-free recipes are lacking: collectively they make up less than 10% of available recipes despite one fifth of users looking for such recipes. Up to 7% of the American population suffers from a form of gluten allergy and up to 20% opt for a gluten-free diet [82], but less than 2.5% of recipes on Food.com are labeled as gluten-free. Users looking for recipes that fit their needs may thus be discouraged from using recipe sites.

6.4 Task

To help home cooks discover delicious and appropriate recipes, we desire a system for *controllable recipe editing*. A user should be able to specify the *base* recipe they would like to edit and the constraint to satisfy. Our model should output a similar but novel recipe that satisfies the constraint. To perform editing on the RecipePairs corpus, we consider a *base* recipe $\mathcal{R}_b = \{N_b; I_b; S_b\}$ comprising name N_b, ingredients I_b, and directions S_b. This recipe does not satisfy a dietary constraint $c \in C$ (e.g. low-fat, dairy-free). The goal of our model is to transform the base recipe into some *target* recipe $\mathcal{R}_t = \{N_t; I_t; S_t\}$ that satisfies c. In this chapter we investigate editing recipes to satisfy a single specified dietary constraint; in future work we aim to extend our method to accommodate multiple restrictions. In our System for Hierarchical Assistive Recipe Editing (**SHARE**), we break the task into two sequential parts: 1) predicting target ingredients I_t given the base name N_b and ingredients I_b alongside dietary constraint c, and 2) generating recipe directions S_t from the new ingredients $\hat{I}_t$.

6.4.1 Why isn't substitution enough?

At first glance, it may seem acceptable to simply replace problematic ingredients with fixed substitutes. To test this, we compiled 269 dietary substitution rules (Table 6.1) from medical websites[2] into the **Rule** baseline that replaces all constraint-violating ingredients in a recipe (e.g. for dairy-free, butter to margarine). In the directions, we replace references to removed ingredients with the substitutes (e.g. beat sugar and butter → beat sugar and *margarine*). For constraint-violating ingredients that do not have a documented substitution, this model removes the ingredient and all references in the text. This method struggles to predict target ingredients (Table 6.2), and cannot account for subtle changes (e.g. cooking techniques) necessary to accommodate new ingredients. Indeed, while *nutritional* impacts of ingredient substitutions are easily inferred, they have intricate effects on recipe structure and palatability. This suggests that recipe editing is a challenging task that cannot be solved by simple rule-based systems.

6.5 SHARE: a Hierarchical Edit Model

We pose *controllable recipe editing* as a supervised learning task: predict a target recipe conditioned on a base recipe and a provided constraint. Our SHARE model consists of two independently trained components for ingredient editing (Section 6.5.1) and step prediction (Section 6.5.2). As ingredient quantities in a recipe may vary based on serving size, we omit exact amounts when editing ingredient lists. When asked to cook recipes edited by our system, home cooks find it relatively easy to decide the amount of each ingredient to use (Section 6.8). To ensure that recipes produced by our systems are 100% safe for cooks with dietary constraints,

[2]e.g. https://www.mayoclinic.org/

we can further filter edited ingredient lists and blacklist inappropriate ingredient tokens from being generated when using our system in the real world. We discuss the instrumentation and impact of filtering and blacklisting modules in Section 6.7.3.

6.5.1 Ingredient Editing

We pose ingredient editing as a multi-label binary classification task: learn the likelihood of an ingredient appearing in the target recipe, conditioned on the base recipe name and ingredients as well as a dietary constraint. First, we embed the category ID and base recipe name via a d-dimensional learned embedding matrix and encode them with a d-dimensional Transformer [210] encoder. We then pass the base recipe ingredient IDs into a d-dimensional Transformer decoder—at each position, each layer of the decoder applies attention over previous positions (self-attention) as well as the encoder outputs. The decoder outputs are projected into an $|\mathcal{I} + 1|$-dimensional vector representing logits for each ingredient in our vocabulary as well as the *eos* token.

A typical transformer decoder predicts ingredients auto-regressively in an ordered list until the *eos* token is encountered: $P(\hat{I}_{t,k}|R_b, c, \hat{I}_{t,<k})$. This strategy penalizes for the order of ingredients, but the ingredients used in a recipe are an un-ordered set—*butter and flour* is the same as *flour and butter*. To remove this order dependence when editing ingredients, we adopt the Set Transformer [179] strategy: we max-pool logits for each ingredient in our output sequence across positions: $P(i \in \hat{I}_t|R_b, c) \;\; \forall \;\; i \in \mathcal{I}$ returning the K ingredients with the highest score overall. We ignore the *eos* logit in the max-pooling; its position denotes the number of predicted ingredients K.

We train the model by maximizing binary cross-entropy (BCE) loss between the pooled ingredient logits and ground truth. We also include a cardinality loss: the BCE loss between the *eos* logit at all time steps compared to the ground truth (1 at position K and 0 otherwise). At inference time, we predict K via Bernoulli sampling from the *eos* logit at each position.

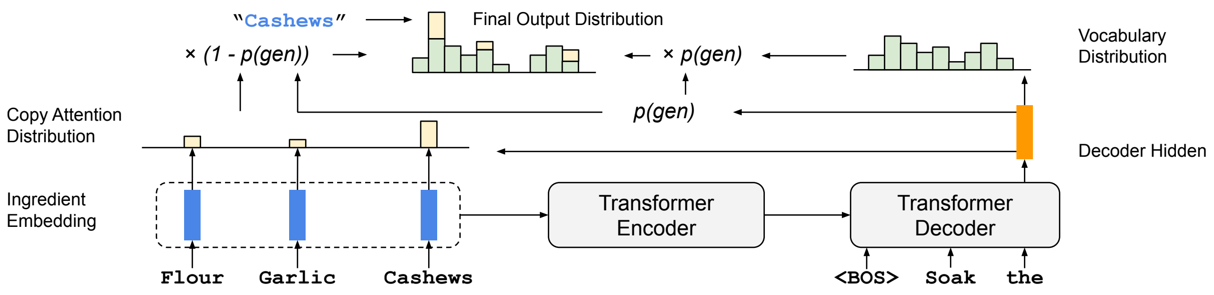


Figure 6.2. SHARE step generation module. The final hidden state at each position is used to calculate p_{gen}, weighing the probability of copying tokens directly from the input ingredients vs. generating from the vocabulary distribution.

6.5.2 Step Generation

We next train a language model that takes our edited set of ingredients as input and generates a new set of recipe steps as a single string (Figure 6.2). This model has no explicit dependency on the base recipe, allowing us to train the model on all 459K recipes from our dataset (excluding evaluation pairs). We follow a conditional language modeling setup: $P(S|\hat{I}_t) = \prod_{k=0}^{K} P(s_k|s_{k-1}\ldots s_0, \hat{I}_t)$.

We represent the sequence of input ingredients $\hat{I}_t$ via their names, joined by a comma for a total of K_{ingr} tokens. Each token w_i is embedded in d dimensions to form the sequence $e_0, \ldots, e_{K_{\text{ingr}}}$, which is then encoded via a Transformer encoder. We obtain a d-dimensional hidden state h_k for each position k in our directions sequence via a Transformer decoder that attends over previous positions as well as the encoder outputs. These hidden states are projected into our vocabulary space $\mathcal{W}$ with a matrix tied to the input embeddings to produce a vocabulary distribution $P(s_k|\hat{I}_t, s_{<k})$ at each position. While this encoder-decoder setup allows us to perform fluent language modeling, it can struggle to produce coherent recipes that properly make use of input ingredients [92]. Thus, we propose to bias our model toward stronger ingredient conditioning by applying copy attention over our input ingredient embeddings.

Ingredient Copy Attention

Our model can directly copy tokens from the input ingredients sequence via a copy attention mechanism [64, 183]. At each position k of our directions, we calculate scaled dot-product attention [210] weights α_k^i over input ingredient tokens i using the final decoder hidden state as query $\mathbf{Q} = h_k \in \mathbb{R}^{1,d}$ and a concatenation of our learned input token embeddings as key $\mathbf{K} = [e_0; \ldots ; e_{K_{\text{ingr}}}] \in \mathbb{R}^{K_{\text{ingr}},d}$: $\alpha_k^i = \text{softmax}(\mathbf{QK}^{\text{T}}/\sqrt{K_{\text{ingr}}})$. We thus learn a distribution representing a chance of copying an ingredient token directly from the input sequence.

At each time-step, our model calculates a generation probability $p_{\text{gen}} \in [0, 1]$ via a learned linear projection from the decoder output h_k: $p_{\text{gen}} = \sigma(W_{\text{gen}}^{\text{T}} h_k + b_{\text{gen}})$ where σ is the sigmoid function. We use p_{gen} as a soft gate to choose between generating the next token from the vocabulary distribution or copying a token from the input ingredients, giving us the final output distribution: $P(w) = p_{\text{gen}} P_{\text{vocab}}(w) + (1 - p_{\text{gen}}) \sum_{i:w_i=w} \alpha_k^i$.

At training time, we compute cross-entropy loss over the predicted recipe directions against the target recipe steps, minimizing the log likelihood of the predicted sequence via the factorized joint distribution: $P(S|\hat{I}_t) = \prod_j^n P(s_k|s_{<k}, \hat{I}_t)$ using the final output distribution.

6.6 Experimental Setup

6.6.1 Baselines

While controllable recipe editing has not been attempted to our knowledge, we adapt state-of-the-art recipe generation models as strong baselines alongside the substitution rule method (**Rule**) described in Section 6.4. Lee et al. [102] fine-tuned a large language model (LM) on Recipe1M+ to extract ingredients from recipe directions and write directions given ingredients: RecipeGPT. We adapt RecipeGPT for multi-task editing (**MT**): 1) predict target ingredients from the category, base recipe name, and base recipe steps by modeling the concatenated sequence $[c; N_b; S_b; I_t]$; and 2) generate target steps from the category, base recipe name, and target recipe ingredients by modeling the concatenated sequence $[c; N_b; I_t; S_t]$.

We next fine-tune RecipeGPT for end-to-end (**E2E**) recipe editing by priming the model [89] with a dietary constraint and base recipe name $[c; N_b]$ (e.g. 'low_fat cheesecake') to generate target ingredients and steps in a single sequence. E2E is twice as efficient as MT and simultaneously predicts ingredients and steps. We also investigate a form of prototype editing in our end-to-end model, by treating the full base recipe as a prototype which needs to be edited to accommodate a constraint (**E2E-P**). While methods for sentence editing use a latent edit vector in addition to a prototype sentence [65], we prime our model with the fully specified base recipe $([c; N_b; I_b; S_b])$ in addition to the category to predict ingredients and steps of an edited recipe.

6.6.2 Evaluation Metrics

We first compare SHARE to baseline models via quantitative experiments on RecipePairs. To measure *fidelity* of edited ingredient lists against ground truth, we calculate Jaccard similarity (IoU) and F1 scores [179], and precision and F1 for insertions and deletions. To measure *step fidelity*, we compare the workflow of two recipes via Normalized Tree Edit distance (NTED) between structured ingredient-cooking-action tree representations [23]. These trees have ingredients as root nodes, with intermediate nodes drawn from a curated set of 259 lemmatised cooking action verbs.

We measure the *fluency* of generated directions via ROUGE-L [119], a metric that assesses *n*-gram overlap between edited and gold recipe texts. We measure *diversity* via the percentage of distinct bigrams (D-2) across edited recipes. To assess whether edited recipes are *appropriate* given the target dietary constraint, we extract all ingredient mentions from recipe directions to flag if ingredients in the edited ingredients list and directions violate the target constraint. We also conduct a human study to critique generated recipes from each model and real-world cooking trials where home cooks follow recipes generated by SHARE to produce dishes (Section 6.8).

Table 6.2. Fidelity of edited ingredient lists when compared to target recipe ingredients, in terms of overall IoU/F1 and insertion/deletion F1 and precision. SHARE creates significantly ($p < 0.05$) higher fidelity edits compared to baselines.

			Insertion		Deletion	
Model	IoU	F1 Score	F1 Score	Precision	F1 Score	Precision
Rule	22.2	33.9	1.2	2.1	18.4	42.0
RecipeGPT (MT)	31.6	45.8	25.6	28.9	69.5	82.4
RecipeGPT (E2E)	29.5	43.2	21.1	26.8	69.5	79.9
RecipeGPT (E2E-P)	30.6	44.7	26.2	29.5	**73.2**	81.0
SHARE	**33.0**	**47.5**	**26.7**	**35.2**	66.1	**83.2**

6.7 Results

6.7.1 RQ 1: How accurately does SHARE edit ingredient lists?

In Table 6.2 we compare edited ingredient lists to ground truth. Ingredient substitution is a challenging task—human-written substitution rules lack the coverage and flexibility required to accurately edit recipes to satisfy dietary constraints. SHARE outperforms all baselines in overall editing accuracy (IoU and F1). We find that correctly adding ingredients is significantly more challenging than removing inappropriate ingredients, with our approach achieving the highest precision in both cases. Baseline LM methods (MT, E2E, E2E-P) predict new ingredients from the vocabulary of English language tokens. SHARE instead separates the ingredient- and step-editing tasks in the hierarchical pipeline, predicting ingredient sets from a known ingredient space. This approach allows us to learn how specific ingredients interact in context of a recipe and dietary restrictions.

6.7.2 RQ 2: How reasonable are the recipe steps generated by SHARE?

We measure recipe direction quality in terms of fluency, diversity, and fidelity (Table 6.3). Here again, the Rule baseline's tendency to simply remove ingredient references not found in human-written substitution rules hurts recipe coherence. Despite other baselines leveraging large pre-trained LMs, no model reports statistically significantly better ROUGE scores than

Table 6.3. Fluency (ROUGE), fidelity (NTED), and diversity (D-2 %) metrics for edited recipe directions. SHARE creates significantly ($p < 0.05$) more structurally similar recipes to the target.

	Fluency: ROUGE ↑	Fidelity: NTED ↓	Diversity: D-2 (%) ↑
Rule	20.3 ± 11.5	0.623 ± 0.072	<u>24.48</u>
RecipeGPT (MT)	**23.3** ± **9.1**	0.618 ± 0.073	7.28
RecipeGPT (E2E)	21.2 ± 8.9	0.621 ± 0.074	10.13
RecipeGPT (E2E-P)	23.0 ± 9.1	0.621 ± 0.068	7.67
SHARE	22.6 ± 7.4	**0.611** ± **0.065**	**13.04**
Paired Data	22.2 ± 7.2	0.614 ± 0.063	13.13
No Copy Attention	20.2 ± 7.5	0.622 ± 0.065	12.04

all alternatives. We confirm observations from Baheti et al. [8] and Majumder et al. [136] that *n*-gram metrics like ROUGE correlate poorly with human judgment (Section 6.8). Our hierarchical ingredient-to-step generation format explicitly decouples the base recipe and edited instructions, allowing SHARE to create more diverse recipes than large LM baselines (+3-5% bigram diversity). We do not require paired data and can train SHARE on all 400K+ recipes from Food.com; we find that a variant trained only on the same paired corpus (**Paired Data**) still out-performs baselines for diversity and fidelity. Thus, independent of corpus size, our hierarchical approach allows our step generator to learn that many dishes can be cooked from the same ingredients—reflecting the many satisfactory ways of editing a recipe to fit a dietary constrain—and addresses a flaw we observe with existing recipe aggregators: a lack of diverse recipes satisfying dietary constraints.

SHARE generates recipes that are not only diverse but also high-fidelity: when comparing workflows between generated recipes and ground truth it significantly out-performs all baselines ($p < 0.05$), producing recipes that are the most structurally similar to gold (lowest average NTED). This reflects how our hierarchical step generator is trained to generate recipes conditioned solely on input ingredients and suggests that it best learns how human cooks make use of a given set of ingredients. We also confirm the importance of our ingredient copy attention module, as a pure Transformer encoder-decoder (**No Copy Attention**) edits recipes with significantly worse structural fidelity.

Table 6.4. Percent of edited ingredients lists violating target hard dietary restrictions (before filtering). SHARE is significantly better at creating appropriate ingredient edits compared to baseline language modeling methods.

Model	Dairy-Free	Gluten-Free	Vegetarian	Overall (Hard Constraints)
Rule	*0.00*	*0.00*	*0.00*	*0.00*
RecipeGPT (MT)	20.55	32.89	10.24	17.23
RecipeGPT (E2E)	39.73	46.05	21.46	30.51
RecipeGPT (E2E-P)	10.96	34.21	4.88	12.43
SHARE	**4.11**	**9.21**	**0.00**	**2.82**

6.7.3 RQ 3: How well does SHARE satisfy dietary constraints?

Previous work in recipe generation has focused on writing recipe steps; we aim to instead create complete recipes satisfying a user's dietary constraints. As some human-written recipes satisfying a soft constraint (e.g. low-sugar) nonetheless judiciously use unhealthy ingredients (e.g. corn syrup), we analyze strict dietary constraints: dairy-free, gluten-free, and vegetarian recipes. Violating such constraints can often result in health risks, so we track the percentage of edited recipes that use prohibited ingredients (from Section 6.3).

SHARE makes significantly more appropriate edits to ingredient lists compared to alternatives (Table 6.4), with less than 3% of edited recipes containing a prohibited ingredient in its ingredients list. Recipe directions, however, are generated as unstructured text by SHARE and LM baselines and can reference problematic ingredients not in the ingredients list. As such, we identify ingredient references in generated steps, and find a 4.5-6.5% violation rate (Table 6.5).

Even if SHARE generates appropriate recipes >95% of the time, a 4.5% chance of generating a potential dangerous recipe edit remains unacceptable for production use. Filtering out prohibited ingredients from the ingredients list helps reduce the incidence of bad references for SHARE, while it has no impact on other LM baselines. This further suggests our approach conditions more strongly on input ingredients when writing steps compared to RecipeGPT-based methods. We can further remove all problematic ingredient references by setting the likelihood of prohibited ingredient sequences to zero during step generation.

Table 6.5. Percent of edited recipe directions referencing ingredients that violate target hard dietary restrictions. SHARE remains the most compliant with requested constraints—and when we remove all inappropriate ingredients from edited lists (+Filter), SHARE is the only model that improves compliance, showing its strong conditioning on input ingredients.

Model	Dairy-Free	Gluten-Free	Vegetarian	Overall (Hard Constraints)
Rule	*0.00*	*0.00*	*0.00*	*0.00*
RecipeGPT (MT)	6.85	22.37	0.49	6.50
+Filter	6.85	22.37	0.49	6.50
RecipeGPT (E2E)	13.70	10.53	0.00	5.08
+Filter	13.70	10.53	0.00	5.08
RecipeGPT (E2E-P)	6.85	14.47	0.98	5.08
+Filter	6.85	14.47	0.98	5.08
SHARE	6.85	14.47	0.00	4.52
+Filter	**4.11**	**11.84**	0.00	**3.39**

6.7.4 Performance vs. Efficiency

In addition to out-performing baseline models for recipe editing, SHARE is much smaller than such models, with 12.5M learnable parameters compared with 124M (~10x larger) for RecipeGPT-based models. Its smaller size confers additional benefits: it takes on average 3.9s to generate a recipe from E2E-P and MT, while it takes on average only 0.9s (**4.3x faster**) for SHARE. Thus, a hierarchical approach to controllable recipe editing accommodates the training of lightweight models for each sub-task that can serve users with acceptable latency.

6.8 Human Studies and Discussion

Procedural texts like recipes indicate a precise series of actions that must be performed with the ingredients at hand to construct a dish. While automatic metrics can help gauge recipe appropriateness, we need human studies to accurately determine whether a recipe—a series of actions performed on a set of ingredients—can be followed. We thus conduct a crowd-sourced human review of recipes generated by each of our models. In addition, we conduct a cooking study with seven cooks—attempting to follow edited instructions and ingredient lists exactly—to

Table 6.6. Issues in edited recipes identified by human evaluators (%): constraint violation, unlisted ingredients, inconsistent naming, lacking step details, lacking ingredient details.

Issue	Gold	Rule	MT	E2E	E2E-P	**SHARE**
Constraint Violation	10.7	15.2	14.3	24.1	8.9	**8.0**
Unlisted Ingredients	<u>7.1</u>	22.3	14.3	12.5	14.3	**7.1**
Inconsistent Naming	4.2	2.1	**0.0**	2.1	10.4	6.3
Lacking Step Details	<u>4.2</u>	14.6	6.3	8.3	8.3	**6.3**
Lacking Ingredient Details	2.1	**0.0**	4.2	4.2	6.3	2.1

assess real-world viability of our system.

6.8.1 Human Evaluation

We randomly sampled eight recipes for each constraint, with each crowd-sourced judge—home cooks familiar with our dietary constraints—given the ground truth and versions edited by each model (total 336 recipes); each recipe was reviewed by two different judges to determine if it violated the target dietary constraint along with providing additional written feedback. We report strong inter-annotator agreement for constraint violation, with a Kendall's Tau [88] value of 0.762. We find a significant difference in the constraint violation rate between models ($p = 0.013$) via a χ^2 test, with SHARE generating the most appropriate outputs across constraints. Some evaluators noted violations of strict constraints, but further inspection revealed these to be safe (e.g. eggs do not trigger dairy allergies). Soft constraint satisfaction remains subjective: a low-carb diet may allow all types of carbs in moderation or disallow specific types (e.g. complex starches). Thus, even some ground truth and Rule baseline recipes can violate a target soft constraint.

77 judges additionally provided written feedback. We identified common themes via qualitative thematic coding [57], as shown in Table 6.6. In 13% of recipe directions, judges noticed references to ingredients that not mentioned in the ingredients list (Unlisted ingredient). In ground truth recipes, this typically consists of optional flavorings (e.g. salt) and garnishes (e.g. fruits), while language-modeling baselines occasionally reference unlisted, substantive ingredients (vegetables and starches). Judges consistently deemed recipes edited by SHARE the

highest quality in terms of appropriateness and references to unlisted ingredients.

The remainder of feedback reflected how stylistic preferences affect perceptions of recipe quality. Several judges felt it confusing when an ingredient was referred to by a synonym or more general phrase (Inconsistent naming: e.g. *cheddar* referred to as *cheese*, or mentions of both *cilantro* and *coriander*). In 5% of recipes (including ground truth), judges asked for more detail in the instructions or ingredients list (Lacking detail). Some judges wondered whether listed ingredients were pre-processed/cooked or raw, and others wanted to know specific utensils and dimensions named for manipulating (e.g. folding) or cutting ingredients.

6.8.2 Cooking Study

Ultimately, one can best judge recipe quality by attempting to cook the dish. We recruited seven home cooks with 3+ years of cooking experience, each tasked to cook 3 random recipes edited by SHARE (total of 21 different recipes covering all seven dietary constraints). Cooks were instructed to follow recipes exactly, recording 1) how complete the recipe was (i.e. missing ingredients/steps); 2) how difficult it was to infer ingredient amounts; 3) overall difficulty of execution; and 4) if the recipe was appropriate for its target category. Table 6.7 shows an example recipe.

No cooks reported difficulties following any recipe. In 90% of the cases, cooks reported using ‘common sense’ or ‘cooking experience’ to ‘easily’ infer ingredient amounts. All cooks agreed that the recipes received were appropriate for their dietary constraint. The most common (19%) complaint among cooks was that the final dish did not resemble the base recipe name. This suggests that there is room for our models to better learn specific textures, shapes, and techniques associated with different food types—while SHARE edits recipes in constraint-respecting and diverse ways, they may surprise user in the process (e.g. editing *curries* into *stews*). Ultimately, we demonstrate that SHARE can successfully create recipes that not only satisfy a specified dietary constraint but are delicious and easily followed by home cooks.

Table 6.7. Example generations from the E2E-P baseline alongside SHARE when asked to make low-calorie versions of Chicken Dijon (original on left). Blue *italics* and red ~~strike through~~ denote added and removed ingredients respectively. The baseline method substitutes olive oil for butter in the ingredients list, but continues to use butter in the instructions.

Base (Chicken Dijon)	Baseline (E2E-P)	SHARE (Ours)
Butter, Chicken, Chicken broth, Dijon mustard, Flour, Light cream, Onion, Pepper, Salt, Wine	Chicken, Dijon mustard, Pepper, Salt, *Dried thyme, Garlic, Lemon juice, Olive oil* ~~Butter, Chicken broth, Flour, Light cream, Onion, Wine~~	Chicken, Dijon mustard, Flour, Pepper, Salt, *Carrot, Garlic, Mushroom, Soy sauce* ~~Butter, Chicken broth, Light cream, Onion, Wine~~
1) Sprinkle the chicken breasts with onion powder, lemon pepper, and salt. 2) Saute the chicken in the butter for 20 min, or cooked through and tender. 3) Remove chicken to a platter and keep warm. 4) Measure the pan juices and enough chicken broth to make 1 cup liquid. 5) Return the stock mixture to the pan and add the wine. 6) Stir together the light cream and the flour till smooth. 7) Add to the broth, cook and stir until smooth, thickened, and bubbly. 8) Stir in the dijon mustard, stir until smooth. 9) Return the chicken to pan, or serve the sauce separately.	1) Combine *lemon juice*, mustard, *garlic*, *thyme*, salt and pepper in a small bowl. 2) Whisk in *butter*. 3) Place chicken in a shallow dish. 4) Pour marinade over chicken. 5) Cover and refrigerate for at least 1 hour. 6) Preheat grill to med-high. 7) Remove chicken from marinade and discard marinade. 8) Grill chicken for 6 minutes on each side or until done.	1) Preheat oven to 350 degrees. 2) Place chicken breasts, breast side up, in a shallow roasting pan. 3) Combine *soy sauce*, mustard, *garlic*, and salt. 4) Brush over both sides of chicken breast. 5) Bake for 40 minutes. 6) Meanwhile, heat a small saucepan over medium heat. 7) Stir in *carrots*, *mushrooms*, and pepper. 8) Cook and stir for 5 minutes. 9) Stir in flour. 10) Pour over chicken breasts, and bake for an additional 10 minutes until chicken is cooked through. 11) Serve chicken with *carrots* and sauce.

6.8.3 Remaining Challenges

As our step predictor depends solely on input ingredients, it can mix cooking techniques to change the form of the edited dish (e.g. casserole in place of pasta). To better serve cooks who crave specific dishes, we hope to extend our work in the future by learning structured representations of recipes [15] and generating recipe instructions as procedural directed graphs [148] instead of raw text. Cooks from different cultural contexts often exhibit different patterns of preferences and requirements in their desired recipes [242], and in the future we hope to extend our work to controllable recipe editing using more subtle traits such as cuisines (e.g. making a Chinese version of meatloaf).

6.9 Conclusion

In this chapter, we propose the task of controllable recipe editing and present a System for Hierarchical Assistive Recipe Editing (SHARE) to tackle this task in the context of dietary constraints. On a novel dataset of 83K pairs of recipes and versions that satisfy dietary constraints, we demonstrate that this is a challenging task that cannot be solved with rule-based ingredient substitutions or straightforward application of existing recipe generators. We show that SHARE can edit recipes to produce coherent, diverse versions that accommodate desired dietary restrictions. Human evaluation and cooking trials also reveal that SHARE produces feasible recipes that can be followed by home cooks to create appealing dishes. In the future, we aim to leverage recipe workflows to generate flow graphs [148] and explore editing recipes to accommodate more complex preferences like cuisines.

6.10 Acknowledgements

This chapter in part is currently being prepared for submission for publication of the material “SHARE: a System for Hierarchical Assistive Recipe Editing” Shuyang Li, Yufei Li, Jianmo Ni, Julian McAuley. The dissertation author was the primary investigator and author of this paper.

Chapter 7

Conclusion and Future Work

In this dissertation, I have presented the primary thrusts of my research into building trustworthy, personalized, and interactive machine learning agents. We have explored probabilistic methods for dialog models to generate informative replies that can be explained via relevant external knowledge. I proposed 1) a question-answering framework for extracting personalized preferences from user utterances and 2) type-centric language modeling algorithms to allow state tracking models to generalize to unseen domains. I have then presented a bot-play framework to enable interactive, conversational recommendation in the absence of dialog transcripts. In the second half of this dissertation we have developed personalized interactive agents to help users discover new cooking recipes that satisfy their dietary needs and restrictions.

We foresee several opportunities for future research to further the themes we have explored in this dissertation. To better adapt to the increasing variety of references and preferences expressed by users, we hope to explore enriching type-centric training corpora with paraphrased facts that reveal relationships between entities, such as from Lu et al. [129]. Furthermore, we foresee that broader usefulness of our type-centric language models as assistive agents—capable of recognizing entities and types in emerging news events and articles, for example—require further study into factual verification and constraints [203] to ensure that we are not surfacing misinformation to users. One possible way to combat misinformation from spurious type associations (e.g. associating a public figure with the type “terrorist”) is to explore the use of intelligent negative sampling during the training process, hopefully improving typing precision.

While we have separately explored extracting user preferences from free-text dialog utterances [114, 117] and incorporating user feedback (regarding specific subjective aspects of an item) in multi-turn conversations [116], in future work we aim to combine the two directions to directly accommodate free-text user feedback for transcript-free conversational recommender systems. We foresee such end-to-end systems as an important step toward building trust and ease of use.

Finally, in the future we aim to combine concepts from our work in generating personalized recipes given seed ingredients [136] with our work in assistive recipe editing [115] to allow personalized recipe generators to iteratively refine recipes to fit the user's specific preferences. I also believe that techniques to enforce internal coherence (i.e. ensuring instructions are internally consistent) are crucial to developing better assistive agents for generating instructional texts (e.g. manuals, cooking recipes, and software APIs). I hope for my continuing research to inspire the development of a wider range of assistive machine learning agents for various facets of daily life.

Bibliography

[1] Oshin Agarwal, Heming Ge, Siamak Shakeri, and Rami Al-Rfou. Knowledge graph based synthetic corpus generation for knowledge-enhanced language model pre-training. In *NAACL-HLT*, pages 3554–3565, 2021. doi: 10.18653/v1/2021.naacl-main.278. URL https://doi.org/10.18653/v1/2021.naacl-main.278.

[2] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Devi Parikh, and Dhruv Batra. VQA: visual question answering. *IJCV*, 123(1):4–31, 2017. doi: 10.1007/s11263-016-0966-6. URL https://doi.org/10.1007/s11263-016-0966-6.

[3] Diego Antognini and Boi Faltings. Fast multi-step critiquing for vae-based recommender systems. In *RecSys*. ACM, 2021. doi: 10.1145/3460231.3474249. URL https://doi.org/10.1145/3460231.3474249.

[4] Diego Antognini, Claudiu Musat, and Boi Faltings. Interacting with explanations through critiquing. In *IJCAI*, pages 515–521. ijcai.org, 2021. doi: 10.24963/ijcai.2021/72. URL https://doi.org/10.24963/ijcai.2021/72.

[5] Jacob Aron. How innovative is apple's new voice assistant, siri?, 2011.

[6] Thomas Bachlechner, Bodhisattwa Prasad Majumder, Huanru Henry Mao, Garrison W. Cottrell, and Julian J. McAuley. Rezero is all you need: Fast convergence at large depth. *CoRR*, abs/2003.04887, 2020. URL https://arxiv.org/abs/2003.04887.

[7] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015. URL http://arxiv.org/abs/1409.0473.

[8] Ashutosh Baheti, Alan Ritter, Jiwei Li, and Bill Dolan. Generating more interesting responses in neural conversation models with distributional constraints. In *EMNLP*, 2018. URL https://aclanthology.info/papers/D18-1431/d18-1431.

[9] Regina Barzilay, Michael Collins, Julia Hirschberg, and Steve Whittaker. The rules behind roles: Identifying speaker role in radio broadcasts. In *AAAI*, 2000. URL http://www.aaai.org/Library/AAAI/2000/aaai00-104.php.

[10] Monika Bednarek. *Evaluation in media discourse: Analysis of a newspaper corpus*. A&C Black, 2006.

[11] Doug Beeferman, William Brannon, and Deb Roy. Radiotalk: A large-scale corpus of talk radio transcripts. In *INTERSPEECH*, 2019. doi: 10.21437/Interspeech.2019-2714.

[12] Ari Biswas, Thai T. Pham, Michael Vogelsong, Benjamin Snyder, and Houssam Nassif. Seeker: Real-time interactive search. In *KDD*, pages 2867–2875. ACM, 2019. doi: 10.1145/3292500.3330733. URL https://doi.org/10.1145/3292500.3330733.

[13] Antoine Bordes, Y-Lan Boureau, and Jason Weston. Learning end-to-end goal-oriented dialog. In *ICLR*. OpenReview.net, 2017. URL https://openreview.net/forum?id=S1Bb3D5gg.

[14] Antoine Bosselut, Asli Çelikyilmaz, Xiaodong He, Jianfeng Gao, Po-Sen Huang, and Yejin Choi. Discourse-aware neural rewards for coherent text generation. In *NAACL-HLT*, 2018. URL https://aclanthology.info/papers/N18-1016/n18-1016.

[15] Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. Simulating action dynamics with neural process networks. In *ICLR*, 2018. URL https://openreview.net/forum?id=rJYFzMZC-.

[16] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In *NeurIPS*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[17] Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. Multiwoz - A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *EMNLP*, pages 5016–5026, 2018. URL https://aclanthology.org/D18-1547/.

[18] Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. Knowledge-based navigation of complex information spaces. In *AAAI*, 1996.

[19] Robin D. Burke, Kristian J. Hammond, and Benjamin C. Young. The findme approach to assisted browsing. *IEEE Expert*, 12(4):32–40, 1997. doi: 10.1109/64.608186.

[20] Giovanni Campagna, Agata Foryciarz, Mehrad Moradshahi, and Monica S. Lam. Zero-shot transfer learning with synthesized data for multi-domain dialogue state tracking. In *ACL*, pages 122–132, 2020. doi: 10.18653/v1/2020.acl-main.12. URL https://doi.org/10.18653/

v1/2020.acl-main.12.

[21] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. In *ICLR*, 2021. URL https://openreview.net/forum?id=5k8F6UU39V.

[22] Justine Cassell and Timothy W. Bickmore. External manifestations of trustworthiness in the interface. *Commun. ACM*, 43(12):50–56, 2000. doi: 10.1145/355112.355123. URL https://doi.org/10.1145/355112.355123.

[23] Minsuk Chang, Leonore V. Guillain, Hyeungshik Jung, Vivian M. Hare, Juho Kim, and Maneesh Agrawala. Recipescape: An interactive tool for analyzing cooking instructions at scale. In *CHI*, 2018. doi: 10.1145/3173574.3174025.

[24] Langzhou Chen, Lori Lamel, Jean-Luc Gauvain, and Gilles Adda. Dynamic language modeling for broadcast news. In *INTERSPEECH*, 2004. URL http://www.isca-speech.org/archive/interspeech_2004/i04_0997.html.

[25] Lu Chen, Boer Lv, Chi Wang, Su Zhu, Bowen Tan, and Kai Yu. Schema-guided multi-domain dialogue state tracking with graph attention neural networks. In *AAAI*, pages 7521–7528, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6250.

[26] Meng Chen, Xiaoyi Jia, Elizabeth Gorbonos, Chnh T Hong, Xiaohui Yu, and Yang Liu. Eating healthier: Exploring nutrition information for healthier recipe recommendation. *Information Processing & Management*, page 102051, 2019.

[27] Yen-Chun Chen, Zhe Gan, Yu Cheng, Jingzhou Liu, and Jingjing Liu. Distilling the knowledge of BERT for text generation. *CoRR*, abs/1911.03829, 2019. URL http://arxiv.org/abs/1911.03829.

[28] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *EMNLP*, 2014. URL http://aclweb.org/anthology/D/D14/D14-1179.pdf.

[29] Eunsol Choi, Omer Levy, Yejin Choi, and Luke Zettlemoyer. Ultra-fine entity typing. In *ACL*, pages 87–96, 2018. doi: 10.18653/v1/P18-1009. URL https://aclanthology.org/P18-1009/.

[30] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. Towards conversational recommender systems. In *KDD*, 2016. doi: 10.1145/2939672.2939746.

[31] Jacob Cohen. A coefficient of agreement for nominal scales. *Educational and psychological measurement*, 20(1):37–46, 1960.

[32] Jordan Cohen. The gale project: A description and an update. In *ASRU*, pages 237–237. IEEE, 2007.

[33] Alison Cook-Sather. Authorizing students' perspectives: Toward trust, dialogue, and change in education. *Educational researcher*, 31(4):3–14, 2002.

[34] Keith S Coulter and Robin A Coulter. Determinants of trust in a service provider: the moderating role of length of relationship. *Journal of services marketing*, 2002.

[35] Nils Dahlbäck, Arne Jönsson, and Lars Ahrenberg. Wizard of oz studies: why and how. In *IUI*, 1993. doi: 10.1145/169891.169968.

[36] Hongliang Dai, Yangqiu Song, and Haixun Wang. Ultra-fine entity typing with weak supervision from a masked language model. In *ACL/IJCNLP*, pages 1790–1799, 2021. doi: 10.18653/v1/2021.acl-long.141. URL https://doi.org/10.18653/v1/2021.acl-long.141.

[37] Karen B. DeSalvo, Richard Olson, and Kellie O. Casavale. Dietary Guidelines for Americans. *JAMA*, 315(5):457–458, 02 2016. ISSN 0098-7484. doi: 10.1001/jama.2015.18396.

[38] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186, 2019. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

[39] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. Wizard of wikipedia: Knowledge-powered conversational agents. In *ICLR*. OpenReview.net, 2019. URL https://openreview.net/forum?id=r1l73iRqKm.

[40] Leyla Dinç and Chris Gastmans. Trust and trustworthiness in nursing: An argument-based literature review. *Nursing inquiry*, 19(3):223–237, 2012.

[41] Xinya Du, Junru Shao, and Claire Cardie. Learning to ask: Neural question generation for reading comprehension. In *ACL*, pages 1342–1352, 2017. doi: 10.18653/v1/P17-1123. URL https://doi.org/10.18653/v1/P17-1123.

[42] Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Kumar Goyal, Peter Ku, and Dilek Hakkani-Tür. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *LREC*, pages 422–428, 2020. URL https://www.aclweb.org/anthology/2020.lrec-1.53/.

[43] Yannick Estève, Thierry Bazillon, Jean-Yves Antoine, Frédéric Béchet, and Jérôme Farinas. The EPAC corpus: Manual and automatic annotations of conversational speech in french broadcast news. In *LREC*, 2010. URL http://www.lrec-conf.org/proceedings/lrec2010/

summaries/650.html.

[44] Norman Fairclough. Discourse representation in media discourse. *Sociolinguistics*, 17(2): 125–139, 1988.

[45] Norman Fairclough and Ruth Wodak. Critical discourse analysis. *Discourse studies: A multidisciplinary introduction*, 2:258–284, 1997.

[46] Angela Fan, Mike Lewis, and Yann N. Dauphin. Hierarchical neural story generation. In *ACL*, 2018. doi: 10.18653/v1/P18-1082. URL https://www.aclweb.org/anthology/P18-1082/.

[47] Angela Fan, Claire Gardent, Chloé Braud, and Antoine Bordes. Using local knowledge graph construction to scale seq2seq models to multi-document inputs. In *EMNLP*, pages 4184–4194, 2019. doi: 10.18653/v1/D19-1428. URL https://doi.org/10.18653/v1/D19-1428.

[48] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. ELI5: long form question answering. In *ACL*, pages 3558–3567, 2019. doi: 10.18653/v1/p19-1346. URL https://doi.org/10.18653/v1/p19-1346.

[49] Thibault Févry, Livio Baldini Soares, Nicholas FitzGerald, Eunsol Choi, and Tom Kwiatkowski. Entities as experts: Sparse memory access with entity supervision. In *EMNLP*. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.400. URL https://doi.org/10.18653/v1/2020.emnlp-main.400.

[50] Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *MRQA at EMNLP*, pages 1–13, 2019. doi: 10.18653/v1/D19-5801. URL https://doi.org/10.18653/v1/D19-5801.

[51] Joseph L Fleiss. Measuring nominal scale agreement among many raters. *Psychological bulletin*, 76(5):378, 1971. URL http://www.wpic.pitt.edu/research/biometrics/Publications/Biometrics%20Archives%20PDF/395-1971%20Fleiss0001.pdf.

[52] Kunihiko Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1(2):119–130, 1988. doi: 10.1016/0893-6080(88)90014-7. URL https://doi.org/10.1016/0893-6080(88)90014-7.

[53] Octavian-Eugen Ganea and Thomas Hofmann. Deep joint entity disambiguation with local neural attention. In *EMNLP*, pages 2619–2629, 2017. doi: 10.18653/v1/d17-1277. URL https://doi.org/10.18653/v1/d17-1277.

[54] Shuyang Gao, Abhishek Sethi, Sanchit Agarwal, Tagyoung Chung, and Dilek Hakkani-Tür.

Dialog state tracking: A neural reading comprehension approach. In *SIGDIAL*, pages 264–273, 2019. doi: 10.18653/v1/W19-5932. URL https://doi.org/10.18653/v1/W19-5932.

[55] Shuyang Gao, Sanchit Agarwal, Tagyoung Chung, Di Jin, and Dilek Hakkani-Tür. From machine reading comprehension to dialogue state tracking: Bridging the gap. *CoRR*, abs/2004.05827, 2020. URL https://arxiv.org/abs/2004.05827.

[56] Albert Gatt and Emiel Krahmer. Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. *J. Artif. Intell. Res.*, 61:65–170, 2018. doi: 10.1613/jair.5477. URL https://doi.org/10.1613/jair.5477.

[57] Graham R Gibbs. Thematic coding and categorizing. *Analyzing qualitative data*, 703: 38–56, 2007.

[58] Augusto Gnisci and Marino Bonaiuto. Grilling politicians: Politicians' answers to questions in television interviews and courtroom examinations. *Journal of language and social psychology*, 22(4):385–413, 2003.

[59] Jeanne P Goldberg, Lindsay A Tanskey, Elizabeth A Sanders, and Marianne Smith Edge. The ific foundation food & health survey 2015: 10-year trends and emerging issues. *JAND*, 117(3):355–357, 2017.

[60] Karthik Gopalakrishnan, Behnam Hedayatnia, Qinglang Chen, Anna Gottardi, Sanjeev Kwatra, Anu Venkatesh, Raefer Gabriel, and Dilek Hakkani-Tür. Topical-chat: Towards knowledge-grounded open-domain conversations. In *INTERSPEECH*, pages 1891–1895. ISCA, 2019. doi: 10.21437/Interspeech.2019-3079. URL https://doi.org/10.21437/Interspeech.2019-3079.

[61] Elizabeth Gorbonos, Yang Liu, and Chính T Hoàng. Nutrec: Nutrition oriented online recipe recommender. In *WI*. IEEE, 2018.

[62] Clinton Gormley and Zachary Tong. *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine*. "O'Reilly Media, Inc.", 2015.

[63] Alex Graves, Santiago Fernández, and Jürgen Schmidhuber. Bidirectional LSTM networks for improved phoneme classification and recognition. In *ICANN*, 2005. doi: 10.1007/11550907_126. URL https://doi.org/10.1007/11550907_126.

[64] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. Incorporating copying mechanism in sequence-to-sequence learning. In *ACL*, 2016. URL http://aclweb.org/anthology/P/P16/P16-1154.pdf.

[65] Kelvin Guu, Tatsunori B. Hashimoto, Yonatan Oren, and Percy Liang. Generating sentences by editing prototypes. *TACL*, 6:437–450, 2018.

[66] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909, 2020. URL https://arxiv.org/abs/2002.08909.

[67] Pentti Haddington. Stance taking in news interviews. *SKY Journal of Linguistics*, 17: 101–142, 2004.

[68] Tatsunori B. Hashimoto, Kelvin Guu, Yonatan Oren, and Percy Liang. A retrieve-and-edit framework for predicting structured outputs. In *NeurIPS*, 2018. URL http://papers.nips.cc/paper/8209-a-retrieve-and-edit-framework-for-predicting-structured-outputs.

[69] Katherine Hawley. Trust and distrust between patient and doctor. *Journal of evaluation in clinical practice*, 21(5):798–801, 2015.

[70] Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gasic. Trippy: A triple copy strategy for value independent neural dialog state tracking. In *SIGDIAL*, pages 35–44, 2020. URL https://www.aclweb.org/anthology/2020.sigdial-1.4/.

[71] Benjamin Heinzerling and Kentaro Inui. Language models as knowledge bases: On entity representations, storage capacity, and paraphrased queries. In *EACL*, pages 1772–1791, 2021. URL https://aclanthology.org/2021.eacl-main.153/.

[72] Matthew Henderson, Blaise Thomson, and Steve J. Young. Word-based dialog state tracking with recurrent neural networks. In *SIGDIAL*, pages 292–299, 2014. doi: 10.3115/v1/w14-4340. URL https://doi.org/10.3115/v1/w14-4340.

[73] John Heritage. Analyzing news interviews: Aspects of the production of talk for an'overhearing'audience. *Handbook of Discourse Analysis, vol. III: Discourse and Dialogue*, 1985.

[74] Carlos M Herrera. A trophic diversity index for presence-absence food data. *Oecologia*, 25(2):187–191, 1976.

[75] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[76] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *ICLR*, 2015. URL http://arxiv.org/abs/1412.6622.

[77] Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. A simple language model for task-oriented dialogue. In *NeurIPS*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/

e946209592563be0f01c844ab2170f0c-Abstract.html.

[78] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *ACL*, pages 328–339. Association for Computational Linguistics, 2018. doi: 10.18653/v1/P18-1031. URL https://aclanthology.org/P18-1031/.

[79] Qian Huang, Zhu Liu, Aaron E. Rosenberg, David C. Gibbon, and Behzad Shahraray. Automated generation of news content hierarchy by integrating audio, video, and text information. In *ICASSP*, 1999. doi: 10.1109/ICASSP.1999.757478. URL https://doi.org/10.1109/ICASSP.1999.757478.

[80] Brian Hutchinson, Bin Zhang, and Mari Ostendorf. Unsupervised broadcast conversation speaker role labeling. In *ICASSP*, 2010. doi: 10.1109/ICASSP.2010.5494958. URL https://doi.org/10.1109/ICASSP.2010.5494958.

[81] Goeric Huybrechts, Thomas Merritt, Giulia Comini, Bartek Perz, Raahil Shah, and Jaime Lorenzo-Trueba. Low-resource expressive text-to-speech using data augmentation. In *ICASSP*, pages 6593–6597. IEEE, 2021. doi: 10.1109/ICASSP39728.2021.9413466. URL https://doi.org/10.1109/ICASSP39728.2021.9413466.

[82] Samuel O Igbinedion, Junaid Ansari, Anush Vasikaran, Felicity N Gavins, Paul Jordan, Moheb Boktor, and Jonathan S Alexander. Non-celiac gluten sensitivity: All wheat attack is not celiac. *World journal of gastroenterology*, 23(40):7201, 2017.

[83] Dongyeop Kang, Anusha Balakrishnan, Pararth Shah, Paul A. Crook, Y-Lan Boureau, and Jason Weston. Recommendation as a communication game: Self-supervised bot-play for goal-oriented dialogue. In *EMNLP*, pages 1951–1961. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1203. URL https://doi.org/10.18653/v1/D19-1203.

[84] Wang-Cheng Kang and Julian McAuley. Self-attentive sequential recommendation. In *ICDM*, 2018. doi: 10.1109/ICDM.2018.00035. URL https://doi.org/10.1109/ICDM.2018.00035.

[85] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP*, pages 6769–6781, 2020. doi: 10.18653/v1/2020.emnlp-main.550. URL https://doi.org/10.18653/v1/2020.emnlp-main.550.

[86] Lauri Karttunen. Syntax and semantics of questions. *Linguistics and philosophy*, 1(1): 3–44, 1977.

[87] Maxime Kayser, Oana-Maria Camburu, Leonard Salewski, Cornelius Emde, Virginie Do, Zeynep Akata, and Thomas Lukasiewicz. e-vil: A dataset and benchmark for natural

language explanations in vision-language tasks. *CoRR*, abs/2105.03761, 2021.

[88] Maurice G Kendall. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93, 1938.

[89] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. CTRL: A conditional transformer language model for controllable generation. *CoRR*, abs/1909.05858, 2019. URL http://arxiv.org/abs/1909.05858.

[90] Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. Unifying question answering and text classification via span extraction. *CoRR*, abs/1904.09286, 2019. URL http://arxiv.org/abs/1904.09286.

[91] Chloé Kiddon, Ganesa Thandavam Ponnuraj, Luke Zettlemoyer, and Yejin Choi. Mise en place: Unsupervised interpretation of instructional recipes. In *EMNLP*, 2015. URL http://aclweb.org/anthology/D/D15/D15-1114.pdf.

[92] Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. Globally coherent text generation with neural checklist models. In *EMNLP*, 2016. URL http://aclweb.org/anthology/D/D16/D16-1032.pdf.

[93] Seokhwan Kim, Michel Galley, R. Chulaka Gunasekara, Sungjin Lee, Adam Atkinson, Baolin Peng, Hannes Schulz, Jianfeng Gao, Jinchao Li, Mahmoud Adada, Minlie Huang, Luis A. Lastras, Jonathan K. Kummerfeld, Walter S. Lasecki, Chiori Hori, Anoop Cherian, Tim K. Marks, Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, and Raghav Gupta. The eighth dialog system technology challenge. *CoRR*, abs/1911.06394, 2019. URL http://arxiv.org/abs/1911.06394.

[94] Sungdong Kim, Sohee Yang, Gyuwan Kim, and Sang-Woo Lee. Efficient dialogue state tracking by selectively overwriting memory. In *ACL*, pages 567–582, 2020. doi: 10.18653/v1/2020.acl-main.53. URL https://doi.org/10.18653/v1/2020.acl-main.53.

[95] Sungdong Kim, Minsuk Chang, and Sang-Woo Lee. Neuralwoz: Learning to collect task-oriented dialogue via model-based simulation. In *ACL*, pages 3704–3717, 2021. doi: 10.18653/v1/2021.acl-long.287. URL https://doi.org/10.18653/v1/2021.acl-long.287.

[96] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. URL http://arxiv.org/abs/1412.6980.

[97] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[98] Yehuda Koren, Robert M. Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. doi: 10.1109/MC.2009.263.

[99] Adarsh Kumar, Peter Ku, Anuj Kumar Goyal, Angeliki Metallinou, and Dilek Hakkani-Tür.

MA-DST: multi-attention-based scalable dialog state tracking. In *AAAI*, pages 8107–8114, 2020. URL https://aaai.org/ojs/index.php/AAAI/article/view/6322.

[100] J. Richard Landis and Gary G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174, 1977. ISSN 0006341X, 15410420. URL http://www.jstor.org/stable/2529310.

[101] Antoine Laurent, Nathalie Camelin, and Christian Raymond. Boosting bonsai trees for efficient features combination: application to speaker role identification. In *INTERSPEECH*, 2014. URL http://www.isca-speech.org/archive/interspeech_2014/i14_0076.html.

[102] Helena H. Lee, Shu Ke, Palakorn Achananuparp, Philips Kokoh Prasetyo, Yue Liu, Ee-Peng Lim, and Lav R. Varshney. Recipegpt: Generative pre-training based cooking recipe generation and evaluation system. In *WWW*, 2020. doi: 10.1145/3366424.3383536.

[103] Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. SUMBT: slot-utterance matching for universal and scalable belief tracking. In *ACL*, pages 5478–5483, 2019. doi: 10.18653/v1/p19-1546. URL https://doi.org/10.18653/v1/p19-1546.

[104] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In *WSDM*, 2020. doi: 10.1145/3336191.3371769. URL https://doi.org/10.1145/3336191.3371769.

[105] Wenqiang Lei, Gangyi Zhang, Xiangnan He, Yisong Miao, Xiang Wang, Liang Chen, and Tat-Seng Chua. Interactive path reasoning on graph for conversational recommendation. In *KDD*, 2020. doi: 10.1145/3394486.3403258. URL https://doi.org/10.1145/3394486.3403258.

[106] Oliver Lemon, Kallirroi Georgila, James Henderson, and Matthew N. Stuttle. An ISU dialogue system exhibiting reinforcement learning of dialogue policies: Generic slot-filling in the TALK in-car system. In *EACL*, 2006. URL https://www.aclweb.org/anthology/E06-2009/.

[107] Leo Leppänen, Myriam Munezero, Mark Granroth-Wilding, and Hannu Toivonen. Data-driven news generation for automated journalism. In *INLG*, 2017. URL https://aclanthology.info/papers/W17-3528/w17-3528.

[108] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, pages 7871–7880, 2020. doi: 10.18653/v1/2020.acl-main.703. URL https://doi.org/10.18653/v1/2020.acl-main.703.

[109] Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *NeurIPS*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

[110] Hanze Li, Scott Sanner, Kai Luo, and Ga Wu. A ranking optimization approach to latent linear critiquing for conversational recommender systems. In *RecSys*, 2020. doi: 10.1145/3383313.3412240.

[111] Jiwei Li, Michel Galley, Chris Brockett, Georgios P. Spithourakis, Jianfeng Gao, and William B. Dolan. A persona-based neural conversation model. In *ACL*. The Association for Computer Linguistics, 2016. doi: 10.18653/v1/p16-1094. URL https://doi.org/10.18653/v1/p16-1094.

[112] Margaret Li, Jason Weston, and Stephen Roller. Acute-eval: Improved dialogue evaluation with optimized questions and multi-turn comparisons, 2019.

[113] Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. Towards deep conversational recommendations. In *NeurIPS*, pages 9748–9758, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/800de15c79c8d840f4e78d3af937d4d4-Abstract.html.

[114] Shuyang Li, Jin Cao, Mukund Sridhar, Henghui Zhu, Shang-Wen Li, Wael Hamza, and Julian J. McAuley. Zero-shot generalization in dialog state tracking through generative question answering. In *EACL*, pages 1063–1074, 2021. URL https://aclanthology.org/2021.eacl-main.91/.

[115] Shuyang Li, Yufei Li, Jianmo Ni, and Julian J. McAuley. SHARE: a system for hierarchical assistive recipe editing. *CoRR*, abs/2105.08185, 2021. URL https://arxiv.org/abs/2105.08185.

[116] Shuyang Li, Bodhisattwa Prasad Majumder, and Julian J. McAuley. Self-supervised bot play for conversational recommendation with rationales. In *TrustNLP at NAACL*, 2022. URL https://arxiv.org/abs/2112.05197.

[117] Shuyang Li, Mukund Sridhar, Chandana Satya Prakash, Jin Cao, Wael Hamza, and Julian J. McAuley. Instilling type knowledge in language models via multi-task QA. In *Findings of NAACL*, 2022. doi: 10.48550/arXiv.2204.13796. URL https://doi.org/10.48550/arXiv.2204.13796.

[118] Yanran Li, Hui Su, Xiaoyu Shen, Wenjie Li, Ziqiang Cao, and Shuzi Niu. Dailydialog: A manually labelled multi-turn dialogue dataset. In *IJCNLP*, 2017.

[119] Chin-Yew Lin and Franz Josef Och. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *ACL*, pages 605–612. ACL, 2004. doi: 10.3115/1218955.1219032. URL https://www.aclweb.org/anthology/P04-1077/.

[120] Zhaojiang Lin, Bing Liu, Seungwhan Moon, Paul A. Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Andrea Madotto, Eunjoon Cho, and Rajen Subba. Leveraging slot descriptions for zero-shot cross-domain dialogue state tracking. *CoRR*, abs/2105.04222, 2021. URL https://arxiv.org/abs/2105.04222.

[121] Xiao Ling and Daniel S. Weld. Fine-grained entity recognition. In *AAAI*. AAAI Press, 2012. URL http://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5152.

[122] Chia-Wei Liu, Ryan Lowe, Iulian Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. How NOT to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *EMNLP*, 2016. doi: 10.18653/v1/d16-1230. URL https://doi.org/10.18653/v1/d16-1230.

[123] Dayiheng Liu, Yeyun Gong, Jie Fu, Yu Yan, Jiusheng Chen, Daxin Jiang, Jiancheng Lv, and Nan Duan. Rikinet: Reading wikipedia pages for natural question answering. In *ACL*, pages 6762–6771, 2020. doi: 10.18653/v1/2020.acl-main.604. URL https://doi.org/10.18653/v1/2020.acl-main.604.

[124] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *ICLR*, 2020.

[125] Yang Liu. Initial study on automatic identification of speaker role in broadcast news speech. In *NAACL-HLT*, 2006. URL https://www.aclweb.org/anthology/N06-2021/.

[126] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692.

[127] Robert L. Logan IV, Nelson F. Liu, Matthew E. Peters, Matt Gardner, and Sameer Singh. Barack's wife hillary: Using knowledge graphs for fact-aware language modeling. In *ACL*, pages 5962–5971, 2019. doi: 10.18653/v1/p19-1598. URL https://doi.org/10.18653/v1/p19-1598.

[128] Gustavo López, Luis Quesada, and Luis A Guerrero. Alexa vs. siri vs. cortana vs. google assistant: a comparison of speech-based natural user interfaces. In *International conference on applied human factors and ergonomics*, pages 241–250. Springer, 2017.

[129] Yinquan Lu, Haonan Lu, Guirong Fu, and Qun Liu. KELM: knowledge enhanced pre-trained language representations with message passing on hierarchical relational graphs.

CoRR, abs/2109.04223, 2021. URL https://arxiv.org/abs/2109.04223.

[130] Yi Luan, Chris Brockett, Bill Dolan, Jianfeng Gao, and Michel Galley. Multi-task learning for speaker-role adaptation in neural conversation models. In *IJCNLP*, 2017. URL https://www.aclweb.org/anthology/I17-1061/.

[131] Joan Lucariello, Amy Kyratzis, and Katherine Nelson. Taxonomic knowledge: What kind and when? *Child development*, 63(4):978–998, 1992.

[132] Kai Luo, Scott Sanner, Ga Wu, Hanze Li, and Hojin Yang. Latent linear critiquing for conversational recommender systems. In *WWW*, 2020. doi: 10.1145/3366423.3380003.

[133] Kai Luo, Hojin Yang, Ga Wu, and Scott Sanner. Deep critiquing for vae-based recommender systems. In *SIGIR*, 2020. doi: 10.1145/3397271.3401091.

[134] Mingyu Derek Ma, Kevin Bowden, JiaQi Wu, Wen Cui, and Marilyn A. Walker. Implicit discourse relation identification for open-domain dialogues. In *ACL*, 2019. doi: 10.18653/v1/p19-1065. URL https://doi.org/10.18653/v1/p19-1065.

[135] Aman Madaan, Amrith Setlur, Tanmay Parekh, Barnabás Póczos, Graham Neubig, Yiming Yang, Ruslan Salakhutdinov, Alan W. Black, and Shrimai Prabhumoye. Politeness transfer: A tag and generate approach. In *ACL*, pages 1869–1881. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.169. URL https://doi.org/10.18653/v1/2020.acl-main.169.

[136] Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian McAuley. Generating personalized recipes from historical user preferences. In *EMNLP*, pages 5975–5981, Nov 2019. doi: 10.18653/v1/D19-1613. URL https://www.aclweb.org/anthology/D19-1613.

[137] Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, Huanru Henry Mao, Sophia Sun, and Julian McAuley. Bernard: A stateful neural open-domain socialbot. *Alexa Prize Proceedings*, 2020.

[138] Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian J. McAuley. Interview: Large-scale modeling of media dialog with discourse patterns and knowledge grounding. In *EMNLP*, pages 8129–8141, 2020. doi: 10.18653/v1/2020.emnlp-main.653. URL https://doi.org/10.18653/v1/2020.emnlp-main.653.

[139] Huanru Henry Mao, Shuyang Li, Julian McAuley, and Garrison W. Cottrell. Speech recognition and multi-speaker diarization of long conversations. *CoRR*, abs/2005.08072, 2020.

[140] Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. Recipe1m+: A dataset for learning cross-modal

embeddings for cooking recipes and food images. *TPAMI*, 2019.

[141] Pierre-Emmanuel Mazaré, Samuel Humeau, Martin Raison, and Antoine Bordes. Training millions of personalized dialogue agents. In *EMNLP*, pages 2775–2779, 2018. doi: 10.18653/v1/d18-1298. URL https://doi.org/10.18653/v1/d18-1298.

[142] Julian J. McAuley, Jure Leskovec, and Dan Jurafsky. Learning attitudes and attributes from multi-aspect reviews. In *ICDM*, 2012. doi: 10.1109/ICDM.2012.110.

[143] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. In *SIGIR*, 2015. doi: 10.1145/2766462.2767755.

[144] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering. *CoRR*, abs/1806.08730, 2018. URL http://arxiv.org/abs/1806.08730.

[145] Sarah McLeod, Ivana Kruijff-Korbayova, and Bernd Kiefer. Multi-task learning of system dialogue act selection for supervised pretraining of goal-oriented dialogue policies. In *SIGDial*. Association for Computational Linguistics, 2019. doi: 10.18653/v1/W19-5947.

[146] Arindam Mitra, Pratyay Banerjee, Kuntal Kumar Pal, Swaroop Mishra, and Chitta Baral. Exploring ways to incorporate additional knowledge to improve natural language commonsense question answering. *CoRR*, abs/1909.08855, 2019. URL http://arxiv.org/abs/1909.08855.

[147] Joshua L. Moore, Shuo Chen, Douglas Turnbull, and Thorsten Joachims. Taste over time: The temporal dynamics of user preferences. In *ISMIR*, 2013. URL http://www.ppgia.pucpr.br/ismir2013/wp-content/uploads/2013/09/220_Paper.pdf.

[148] Shinsuke Mori, Hirokuni Maeta, Yoko Yamakata, and Tetsuro Sasada. Flow graph corpus from recipe texts. In *LREC*, 2014. URL http://www.lrec-conf.org/proceedings/lrec2014/summaries/763.html.

[149] Nikola Mrksic, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve J. Young. Neural belief tracker: Data-driven dialogue state tracking. In *ACL*, pages 1777–1788, 2017. doi: 10.18653/v1/P17-1163. URL https://doi.org/10.18653/v1/P17-1163.

[150] Jianmo Ni and Julian McAuley. Personalized review generation by expanding phrases and attending on aspect-aware representations. In *ACL*, 2018. URL https://aclanthology.info/papers/P18-2112/p18-2112.

[151] Jianmo Ni, Zachary C. Lipton, Sharad Vikram, and Julian McAuley. Estimating reactions and recommending products with generative models of reviews. In *IJCNLP*, 2017. URL

https://aclanthology.info/papers/I17-1079/i17-1079.

[152] Jianmo Ni, Jiacheng Li, and Julian J. McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *EMNLP*, 2019. doi: 10.18653/v1/D19-1018.

[153] Tong Niu and Mohit Bansal. Polite dialogue generation without parallel data. *TACL*, 6: 373–389, 2018. URL https://transacl.org/ojs/index.php/tacl/article/view/1424.

[154] Jekaterina Novikova, Ondrej Dusek, Amanda Cercas Curry, and Verena Rieser. Why we need new evaluation metrics for NLG. In *EMNLP*, 2017. doi: 10.18653/v1/d17-1238. URL https://doi.org/10.18653/v1/d17-1238.

[155] Yasumasa Onoe and Greg Durrett. Interpretable entity representations through large-scale typing. In *Findings of EMNLP*, pages 612–624, 2020. doi: 10.18653/v1/2020.findings-emnlp.54. URL https://doi.org/10.18653/v1/2020.findings-emnlp.54.

[156] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *ACL*, 2002. doi: 10.3115/1073083.1073135. URL https://www.aclweb.org/anthology/P02-1040/.

[157] German Ignacio Parisi, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019. doi: 10.1016/j.neunet.2019.01.012. URL https://doi.org/10.1016/j.neunet.2019.01.012.

[158] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization. In *ICLR*, 2018. URL https://openreview.net/forum?id=HkAClQgA-.

[159] Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. SOLOIST: few-shot task-oriented dialog with A single pre-trained auto-regressive model. *CoRR*, abs/2005.05298, 2020. URL https://arxiv.org/abs/2005.05298.

[160] Matthew E. Peters, Mark Neumann, Robert L. Logan IV, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. Knowledge enhanced contextual word representations. In *EMNLP*, pages 43–54, 2019. doi: 10.18653/v1/D19-1005. URL https://doi.org/10.18653/v1/D19-1005.

[161] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick S. H. Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander H. Miller. Language models as knowledge bases? In *EMNLP*, pages 2463–2473, 2019. doi: 10.18653/v1/D19-1250. URL https://doi.org/10.18653/v1/D19-1250.

[162] Paul Placeway, Scotte Chen, Maxine Eskénazi, Uday Jain, Vipul N. Parikh, Bhiksha Raj,

Mosur Ravishankar, Rogério Rosenfeld, Kristie Seymore, Matthew A. Siegler, Richard M. Stern, and Eric H. Thayer. The 1996 hub-4 sphinx-3 system. In *Proc. DARPA Speech recognition workshop*, volume 97. Citeseer, 1997.

[163] Yujia Qin, Yankai Lin, Ryuichi Takanobu, Zhiyuan Liu, Peng Li, Heng Ji, Minlie Huang, Maosong Sun, and Jie Zhou. ERICA: improving entity and relation understanding for pre-trained language models via contrastive learning. In *ACL*, pages 3350–3363. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.260. URL https://doi.org/10.18653/v1/2021.acl-long.260.

[164] Lingyun Qiu and Izak Benbasat. Evaluating anthropomorphic product recommendation agents: A social relationship perspective to designing information systems. *J. Manag. Inf. Syst.*, 25(4):145–182, 2009.

[165] Massimo Quadrana, Paolo Cremonesi, and Dietmar Jannach. Sequence-aware recommender systems. In *UMAP*, 2018. doi: 10.1145/3209219.3209270. URL https://doi.org/10.1145/3209219.3209270.

[166] Dragomir R. Radev, Hong Qi, Harris Wu, and Weiguo Fan. Evaluating web-based question answering systems. In *LREC*, 2002. URL http://www.lrec-conf.org/proceedings/lrec2002/pdf/301.pdf.

[167] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI*, 2018.

[168] Alec Radford, Jeffrey Wu, Dario Amodei, Daniela Amodei, Jack Clark, Miles Brundage, and Ilya Sutskever. Better language models and their implications. *OpenAI Blog*, 2019. URL https://openai.com/blog/better-language-models/.

[169] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

[170] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*, 21:140:1–140:67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

[171] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. In *ACL*, pages 784–789, 2018. doi: 10.18653/v1/P18-2124. URL https://www.aclweb.org/anthology/P18-2124/.

[172] Osman Ramadan, Pawel Budzianowski, and Milica Gasic. Large-scale multi-domain belief tracking with knowledge sharing. In *ACL*, pages 432–437, 2018. doi: 10.18653/v1/P18-2069. URL https://www.aclweb.org/anthology/P18-2069/.

[173] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, and John Riedl. Getting to know you: learning new user preferences in recommender systems. In *IUI*, 2002. doi: 10.1145/502716.502737. URL https://doi.org/10.1145/502716.502737.

[174] Ehud Reiter and Robert Dale. Building applied natural language generation systems. *Nat. Lang. Eng.*, 3(1):57–87, 1997. doi: 10.1017/S1351324997001502. URL https://doi.org/10.1017/S1351324997001502.

[175] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In *UAI*, 2009.

[176] Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, and Jason Weston. Recipes for building an open-domain chatbot. In *EACL*, pages 300–325. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.eacl-main.24. URL https://doi.org/10.18653/v1/2021.eacl-main.24.

[177] Ahmad Sakor, Kuldeep Singh, Anery Patel, and Maria-Esther Vidal. Falcon 2.0: An entity and relation linking tool over wikidata. In *CIKM*, pages 3141–3148. ACM, 2020. doi: 10.1145/3340531.3412777. URL https://doi.org/10.1145/3340531.3412777.

[178] Gerard Salton and Chris Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manag.*, 24(5):513–523, 1988. doi: 10.1016/0306-4573(88)90021-0. URL https://doi.org/10.1016/0306-4573(88)90021-0.

[179] Amaia Salvador, Michal Drozdzal, Xavier Giró-i-Nieto, and Adriana Romero. Inverse cooking: Recipe generation from food images. In *CVPR*, 2019. doi: 10.1109/CVPR.2019.01070.

[180] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter. *CoRR*, abs/1910.01108, 2019. URL http://arxiv.org/abs/1910.01108.

[181] Chinnadhurai Sankar, Sandeep Subramanian, Chris Pal, Sarath Chandar, and Yoshua Bengio. Do neural dialog systems use the conversation history effectively? an empirical study. In *ACL*, 2019. doi: 10.18653/v1/p19-1004. URL https://doi.org/10.18653/v1/p19-1004.

[182] Suvash Sedhain, Hung Bui, Jaya Kawale, Nikos Vlassis, Branislav Kveton, Aditya Krishna Menon, Trung Bui, and Scott Sanner. Practical linear models for large-scale one-class collaborative filtering. In *IJCAI*, 2016.

[183] Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization

with pointer-generator networks. In *ACL*, 2017. doi: 10.18653/v1/P17-1099. URL https://doi.org/10.18653/v1/P17-1099.

[184] Abigail See, Stephen Roller, Douwe Kiela, and Jason Weston. What makes a good conversation? how controllable attributes affect human judgments. In *NAACL-HLT*, 2019. URL https://www.aclweb.org/anthology/N19-1170/.

[185] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *ACL*, 2016. URL http://aclweb.org/anthology/P/P16/P16-1162.pdf.

[186] Pararth Shah, Dilek Hakkani-Tür, Gökhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry P. Heck. Building a conversational agent overnight with dialogue self-play. *CoRR*, abs/1801.04871, 2018. URL http://arxiv.org/abs/1801.04871.

[187] Siamak Shakeri, Cícero Nogueira dos Santos, Henghui Zhu, Patrick Ng, Feng Nan, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. End-to-end synthetic data generation for domain adaptation of question answering systems. In *EMNLP*, pages 5445–5460, 2020. doi: 10.18653/v1/2020.emnlp-main.439. URL https://doi.org/10.18653/v1/2020.emnlp-main.439.

[188] Yuanlong Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. Generating high-quality and informative conversation responses with sequence-to-sequence models. In *EMNLP*, 2017. doi: 10.18653/v1/d17-1235. URL https://doi.org/10.18653/v1/d17-1235.

[189] Richard Shepherd, Claire M Paisley, Paul Sparks, Annie S Anderson, Susan Eley, and Mike EJ Lean. Constraints on dietary choice: the role of income. *Nutrition & Food Science*, 1996.

[190] Heung-Yeung Shum, Xiaodong He, and Di Li. From eliza to xiaoice: challenges and opportunities with social chatbots. *Frontiers Inf. Technol. Electron. Eng.*, 19(1):10–26, 2018. doi: 10.1631/FITEE.1700826. URL https://doi.org/10.1631/FITEE.1700826.

[191] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. In *EMNLP (Findings)*, pages 3784–3803, 2021. URL https://aclanthology.org/2021.findings-emnlp.320.

[192] Sameer Singh, Amarnag Subramanya, Fernando Pereira, and Andrew McCallum. Wikilinks: A large-scale cross-document coreference corpus labeled via links to Wikipedia. Technical Report UM-CS-2012-015, University of Massachusetts, Amherst, 2012. URL https://web.cs.umass.edu/publication/docs/2012/UM-CS-2012-015.pdf.

[193] Rashmi R. Sinha and Kirsten Swearingen. The role of transparency in recommender systems. In *CHI*, 2002. doi: 10.1145/506443.506619.

[194] Brian G Southwell, Emily A Thorson, and Laura Sheble. *Misinformation and mass audiences*. University of Texas Press, 2018.

[195] Georgios P. Spithourakis and Sebastian Riedel. Numeracy for language models: Evaluating and improving their ability to predict numbers. In *ACL*, pages 2104–2115, 2018. doi: 10.18653/v1/P18-1196. URL https://www.aclweb.org/anthology/P18-1196/.

[196] Stephanie M Strassel. Linguistic resources for effective, affordable, reusable speech-to-text. In *LREC*, 2004.

[197] Tianxiang Sun, Yunfan Shao, Xipeng Qiu, Qipeng Guo, Yaru Hu, Xuanjing Huang, and Zheng Zhang. Colake: Contextualized language and knowledge embedding. In *COLING*. International Committee on Computational Linguistics, 2020. doi: 10.18653/v1/2020.coling-main.327. URL https://doi.org/10.18653/v1/2020.coling-main.327.

[198] Yu Sun, Shuohuan Wang, Yu-Kun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. ERNIE 2.0: A continual pre-training framework for language understanding. In *AAAI*. AAAI Press, 2020. URL https://ojs.aaai.org/index.php/AAAI/article/view/6428.

[199] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112, 2014. URL https://proceedings.neurips.cc/paper/2014/hash/a14ac55a4f27472c5d894ec1c3c743d2-Abstract.html.

[200] ChunYuen Teng, Yu-Ru Lin, and Lada A. Adamic. Recipe recommendation using ingredient networks. In *WebSci*, 2012. doi: 10.1145/2380718.2380757. URL https://doi.org/10.1145/2380718.2380757.

[201] Raghuveer Thirukovalluru, Mukund Sridhar, Dung Thai, Shruti Chanumolu, Nicholas Monath, Sankaranarayanan Ananthakrishnan, and Andrew McCallum. Knowledge informed semantic parsing for conversational question answering. In *RepL4NLP*, pages 231–240, Online, 2021. doi: 10.18653/v1/2021.repl4nlp-1.24. URL https://aclanthology.org/2021.repl4nlp-1.24.

[202] Blaise Thomson and Steve J. Young. Bayesian update of dialogue state: A POMDP framework for spoken dialogue systems. *Comput. Speech Lang.*, 24(4):562–588, 2010. doi: 10.1016/j.csl.2009.07.003. URL https://doi.org/10.1016/j.csl.2009.07.003.

[203] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and verification. In *NAACL-HLT*, pages 809–819. Association for Computational Linguistics, 2018. doi: 10.18653/v1/n18-1074. URL https://doi.org/10.18653/v1/n18-1074.

[204] Nava Tintarev and Judith Masthoff. Designing and evaluating explanations for recommender systems. In *Recommender Systems Handbook*, pages 479–510. Springer, 2011.

[205] Ya-Min Tseng, Yi-Ting Huang, Meng Chang Chen, and Yeali S. Sun. Generating comprehension questions using paraphrase. In *Technologies and Applications of Artificial Intelligence, 19th International Conference, TAAI 2014, Taipei, Taiwan, November 21-23, 2014. Proceedings*, pages 310–321, 2014. doi: 10.1007/978-3-319-13987-6_29. URL https://doi.org/10.1007/978-3-319-13987-6_29.

[206] Amos Tversky and Itamar Simonson. Context-dependent preferences. *Management Science*, 39(10):1179–1189, 1993. ISSN 00251909, 15265501.

[207] Mayumi Ueda, Mari Takahata, and Shinsuke Nakajima. User's food preference extraction for personalized cooking recipe recommendation. In *SPIM*, 2011. URL http://dl.acm.org/citation.cfm?id=2887675.2887686.

[208] Teun A van Dijk. *Discourse and communication: New approaches to the analysis of mass media discourse and communication*, volume 10. Walter de Gruyter, 2011.

[209] Suus MJ Van Hekken and Wim Roelofsen. More questions than answers: A study of question–answer sequences in a naturalistic setting. *Journal of Child Language*, 9(2): 445–460, 1982.

[210] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[211] Jesse Vig, Shilad Sen, and John Riedl. Tagsplanations: explaining recommendations using tags. In *IUI*, 2009. doi: 10.1145/1502650.1502661.

[212] Denny Vrandecic. Wikidata: a new platform for collaborative data collection. In *WWW*, pages 1063–1064. ACM, 2012. doi: 10.1145/2187980.2188242. URL https://doi.org/10.1145/2187980.2188242.

[213] Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. Do NLP models know numbers? probing numeracy in embeddings. In *EMNLP*, pages 5306–5314, 2019. doi: 10.18653/v1/D19-1534. URL https://doi.org/10.18653/v1/D19-1534.

[214] Mengting Wan and Julian J. McAuley. Item recommendation on monotonic behavior chains. In *RecSys*, 2018. doi: 10.1145/3240323.3240369.

[215] 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. In *ICLR*, 2019. URL https://openreview.net/forum?id=rJ4km2R5t7.

[216] Bingning Wang, Kang Liu, and Jun Zhao. Inner attention based recurrent neural

networks for answer selection. In *ACL*, 2016. doi: 10.18653/v1/p16-1122. URL https://doi.org/10.18653/v1/p16-1122.

[217] Wen Wang, Sibel Yaman, Kristin Precoda, and Colleen Richey. Automatic identification of speaker role and agreement/disagreement in broadcast conversation. In *ICASSP*, 2011. doi: 10.1109/ICASSP.2011.5947618. URL https://doi.org/10.1109/ICASSP.2011.5947618.

[218] Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *TACL*, 9:176–194, 2021. URL https://transacl.org/ojs/index.php/tacl/article/view/2447.

[219] Zhuoran Wang and Oliver Lemon. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *SIGDIAL*, pages 423–432, 2013. URL https://www.aclweb.org/anthology/W13-4067/.

[220] Pontus Wärnestål. Modeling a dialogue strategy for personalized movie recommendations. In *Beyond Personalization Workshop*, pages 77–82, 2005.

[221] Elda Weizman. *Positioning in media dialogue: Negotiating roles in the news interview*, volume 3. John Benjamins Publishing, 2008.

[222] Ronald J. Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2):270–280, 1989. doi: 10.1162/neco.1989.1.2.270. URL https://doi.org/10.1162/neco.1989.1.2.270.

[223] Sam Witteveen and Martin Andrews. Paraphrasing with large language models. In *NGT@EMNLP-IJCNLP*, 2019. doi: 10.18653/v1/D19-5623. URL https://doi.org/10.18653/v1/D19-5623.

[224] Richard Wrangham. *Catching fire: how cooking made us human*. Basic Books, 2009.

[225] Stephen J. Wright. Coordinate descent algorithms. *Math. Program.*, 151(1):3–34, 2015. doi: 10.1007/s10107-015-0892-3. URL https://doi.org/10.1007/s10107-015-0892-3.

[226] Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. Transferable multi-domain state generator for task-oriented dialogue systems. In *ACL*, pages 808–819, 2019. doi: 10.18653/v1/p19-1078. URL https://doi.org/10.18653/v1/p19-1078.

[227] Ga Wu, Kai Luo, Scott Sanner, and Harold Soh. Deep language-based critiquing for recommender systems. In *RecSys*, 2019. doi: 10.1145/3298689.3347009.

[228] Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer.

Scalable zero-shot entity linking with dense entity retrieval. In *EMNLP*, pages 6397–6407, 2020. doi: 10.18653/v1/2020.emnlp-main.519. URL https://doi.org/10.18653/v1/2020.emnlp-main.519.

[229] Jingjing Xu, Xuancheng Ren, Yi Zhang, Qi Zeng, Xiaoyan Cai, and Xu Sun. A skeleton-based model for promoting coherence among sentences in narrative story generation. In *EMNLP*, 2018. URL https://aclanthology.info/papers/D18-1462/d18-1462.

[230] Puyang Xu and Qi Hu. An end-to-end approach for handling unknown slot values in dialogue state tracking. In *ACL*, pages 1448–1457, 2018. doi: 10.18653/v1/P18-1134. URL https://www.aclweb.org/anthology/P18-1134/.

[231] Ikuya Yamada, Koki Washio, Hiroyuki Shindo, and Yuji Matsumoto. Global entity disambiguation with pretrained contextualized embeddings of words and entities. *CoRR*, abs/1909.00426, 2019. URL https://arxiv.org/abs/1909.00426.

[232] Ikuya Yamada, Akari Asai, Hiroyuki Shindo, Hideaki Takeda, and Yuji Matsumoto. LUKE: deep contextualized entity representations with entity-aware self-attention. In *EMNLP*. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.523. URL https://doi.org/10.18653/v1/2020.emnlp-main.523.

[233] Ryosuke Yamanishi, Naoki Shino, Yoko Nishihara, Junichi Fukumoto, and Aya Kaizaki. Alternative-ingredient recommendation based on co-occurrence relation on recipe database. In *KES*, 2015. doi: 10.1016/j.procs.2015.08.138. URL https://doi.org/10.1016/j.procs.2015.08.138.

[234] Kevin Yang and Dan Klein. FUDGE: controlled text generation with future discriminators. In *NAACL-HLT*, pages 3511–3535. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.276. URL https://doi.org/10.18653/v1/2021.naacl-main.276.

[235] Pengcheng Yang, Xu Sun, Wei Li, Shuming Ma, Wei Wu, and Houfeng Wang. SGM: sequence generation model for multi-label classification. In *COLING*, pages 3915–3926, 2018. URL https://aclanthology.org/C18-1330/.

[236] Yunyi Yang, Yunhao Li, and Xiaojun Quan. UBAR: towards fully end-to-end task-oriented dialog systems with GPT-2. *CoRR*, abs/2012.03539, 2020. URL https://arxiv.org/abs/2012.03539.

[237] Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. Reference-aware language models. In *EMNLP*, 2017. URL https://aclanthology.info/papers/D17-1197/d17-1197.

[238] Lili Yao, Nanyun Peng, Ralph M. Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. Plan-and-write: Towards better automatic storytelling. *CoRR*, abs/1811.05701, 2018.

URL http://arxiv.org/abs/1811.05701.

[239] Yuan Yao, Deming Ye, Peng Li, Xu Han, Yankai Lin, Zhenghao Liu, Zhiyuan Liu, Lixin Huang, Jie Zhou, and Maosong Sun. Docred: A large-scale document-level relation extraction dataset. In *ACL*, pages 764–777, 2019. doi: 10.18653/v1/p19-1074. URL https://doi.org/10.18653/v1/p19-1074.

[240] Yang You, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. In *ICLR*. OpenReview.net, 2020. URL https://openreview.net/forum?id=Syx4wnEtvH.

[241] Jianguo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wan, Philip S. Yu, Richard Socher, and Caiming Xiong. Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking. *CoRR*, abs/1910.03544, 2019. URL http://arxiv.org/abs/1910.03544.

[242] Qing Zhang, Christoph Trattner, Bernd Ludwig, and David Elsweiler. Understanding cross-cultural visual food tastes with online recipe platforms. In *ICWSM*, 2019. URL https://aaai.org/ojs/index.php/ICWSM/article/view/3270.

[243] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too? In *ACL*, 2018. doi: 10.18653/v1/P18-1205. URL https://www.aclweb.org/anthology/P18-1205/.

[244] Taolin Zhang, Chengyu Wang, Nan Hu, Minghui Qiu, Chengguang Tang, Xiaofeng He, and Jun Huang. DKPLM: decomposable knowledge-enhanced pre-trained language model for natural language understanding. *CoRR*, abs/2112.01047, 2021. URL https://arxiv.org/abs/2112.01047.

[245] Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. Dialogpt: Large-scale generative pre-training for conversational response generation. *CoRR*, abs/1911.00536, 2019. URL http://arxiv.org/abs/1911.00536.

[246] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*, 2014. doi: 10.1145/2600428.2609579.

[247] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W. Bruce Croft. Towards conversational search and recommendation: System ask, user respond. In *CIKM*, 2018. doi: 10.1145/3269206.3271776.

[248] Tiancheng Zhao, Ran Zhao, and Maxine Eskénazi. Learning discourse-level diversity

for neural dialog models using conditional variational autoencoders. In *ACL*, 2017. doi: 10.18653/v1/P17-1061. URL https://doi.org/10.18653/v1/P17-1061.

[249] Kun Zhou, Wayne Xin Zhao, Shuqing Bian, Yuanhang Zhou, Ji-Rong Wen, and Jingsong Yu. Improving conversational recommender systems via knowledge graph based semantic fusion. In *KDD*, pages 1006–1014. ACM, 2020.

[250] Li Zhou and Kevin Small. Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering. *CoRR*, abs/1911.06192, 2019. URL http://arxiv.org/abs/1911.06192.