UNIVERSITY OF CALIFORNIA
SANTA CRUZ

MODELING KEY NARRATIVE ELEMENTS FOR STORY UNDERSTANDING AND GENERATION

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

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE

by

Faeze Brahman

March 2022

The Dissertation of Faeze Brahman
is approved:

Professor Snigdha Chaturvedi, Chair

Professor Jeffrey Flanigan

Professor Seshadhri Comandur

Peter F. Biehl
Vice Provost and Dean of Graduate Studies

Contents

List of Figures

List of Tables

University of California Santa Cruz

Abstract

Modeling Key Narrative Elements for Story Understanding and Generation

Faeze Brahman

Narratives are central to how humans communicate, reason, and make sense of their experiences. They are a rich source of day-to-day knowledge and contain many social and moral norms. Therefore, narratives can be used to instill human-like communication, commonsense knowledge, and reasoning capabilities in machines. Towards this goal, Artificial Intelligence (AI) and Natural Language Processing (NLP) community, in particular, have long been interested in *understanding* and *generating* narratives. While understanding and generating narratives are natural for most people, it remains an elusive goal for AI and NLP systems, in part because it requires an understanding of key narrative elements and how they evolve and drive the story forward.

In this dissertation, we approach these challenges by designing new tasks as well as resources specifically aimed toward advancing narrative *understanding* and *generation* models. We propose to model and incorporate narrative elements that contribute to a good story, such as *plot*, *characters*, and *emotions*. We also introduce new modeling frameworks to incorporate user specifications in the form of *narrative elements* into the generation process.

First, we model the *plot* of the story via user-provided *cue phrases* as a narrative element in an interactive generation framework. We present two content-inducing approaches based on the Transformer Network (Vaswani et al., 2017) for interactively incorporating cue phrases when automatically generating stories. Next, we present to model the *evolution of emotions* in neural story generation. We develop methods to generate stories that adhere to desired *emotion arcs*. We specifically designed two

emotion-consistency rewards in a Reinforcement Learning framework to regularize the story generation. We then study *character*(*s*) as narrative elements. We introduce a new dataset, `LiSCU`, and tasks for assessing machines' understanding of characters which is of utmost importance for understanding narratives. Lastly, we focus on entities (and not just characters) as a narrative element. We propose a real-world application of generating narratives about entities grounded in world knowledge. Towards this, we present a new dataset EntDeGen, ranking-based generation models, and an evaluation metric of factuality.

DEDICATION

To my dearest parents, Farzane and Ali,

and to my beloved husband, Ehsan!

Acknowledgements

My Ph.D. journey has been full of ups and downs, and in no way have I ever been able to complete a successful Ph.D. if it weren't for all the people who supported me during this path.

First and foremost, I would like to extend my special thanks to my advisor, Snigdha Chaturvedi, for her exceptional mentorship and support, teaching me how to conduct quality research, and always challenging me to grow.

I also thank Seshadhri Comandur and Jeffrey Flanigan for serving on my committee and for their insightful comments and discussions. I would like to thank BSOE graduate advising team— Alicia Haley and Tracie Tucker for all their great help throughout my Ph.D. program.

I have been fortunate to work with several wonderful collaborators and workshop co-organizers during my Ph.D. I want to thank (in no particular order): Chao Zhao, Somnath Basu Roy Chowdhury, Vered Shwartz, Yejin Choi, Oyvind Tafjord, Rachel Rudinger, Mrinmaya Sachan, Meng Huang, Michel Galley, Jianfeng Gao, Baolin Peng, Sudha Rao, Bill Dolan, Alexandru Petrusca, Tenghao Huang, Anvesh Rao, Mohit Iyyer, Elizabeth Clark, Nader Akoury, and Lara J. Martin. I am also lucky to have many friends, both close and distant, who have always been there and helped me stay positive during research endeavors—Javane Rostampoor, Sama Kebrity, Fatemeh Mirzaei, Sara Bonabi, Rojin Safavi, Holakou Rahmanian, Ehsan Amid, and Sahar Derakhshani.

Special thanks to Vered Shwartz and Yejin Choi for hosting me at AI2 during the most enjoyable internship. Their mentorship helped me to develop a systematic approach to

research. I also thank Antoine Bosselut, Peter Clark, Daniel Khashabi, Oyvind Tafjord, and several others from AI2 for the fruitful research discussions. I thank Michel Galley and Jianfeng Gao at Microsoft Research, where I interned in Summer 2021 and wrote the last chapter of this thesis.

Last but not least, I am forever grateful to my family: my parents, Farzane and Ali, for their endless love and support, without whom I wouldn't be where I am now, and my sister, Azade, for always encouraging me to pursue my dreams, and my brother, Ehsan, for distracting me with fun things. My special thank goes to my beloved husband, Ehsan, for his unwavering emotional support and positive energy all along, for bringing meaning to my life and for prioritizing my success.

— March 2022, California.

Chapter 1

Introduction

Human life is narratively rooted.

Peter L. Berger

In addition to being the primary content of the media we consume daily, and hourly, narratives play an integral role in human cognition. They are central to reasoning and sense-making about everyday experiences (Anderson, 2015) and allow us to communicate, express emotions, share knowledge, and shape our perspective of the world (McKee, 2003). For example, when presenting, people use narrative to support their points and make their speeches more compelling. They provide examples or points of view in a way that resonates with an audience. Listeners connect with stories emotionally, associating their feelings with their learning.

By definition, a narrative is a series of related events in a cohesive and logical format (Prince, 1973). Narratives are a rich source of day-to-day knowledge and contain many social and moral norms, making them a great resource to instill human-like communication, commonsense knowledge, and reasoning capabilities in machines. Towards this goal, artificial intelligence (AI) and natural language processing (NLP) community, in particular, have for long been interested in *understanding*, and *generating* narratives.

There are key narrative elements that contribute to a good story such as *plot*, *characters*, *emotions*, *goals* etc. *Understanding* narratives requires understanding of characters beyond

named entities and reasoning about various aspects of characters such as roles, personalities, relationships, intents, consequences of actions, etc. (aka social commonsense inferences) (Piper et al., 2021). Automatically *generating* narratives requires composing a causally related and logically coherent sequence of events involving characters, given a topic and a set of specifications. To produce a coherent story, an AI system has to have a lot of world and commonsense knowledge about the social dynamics of characters and actions and causal relations within stories. When composing a story, human writers start with a rough sketch of what key events and the main character(s) are involved. As they unfold the story, they must keep track of these elements while weaving together the characters, their actions and emotions, and events in a coherent and consistent way. Despite the significance of explicitly modeling these narrative elements for *understanding* and automatically *generating* stories, there are limited NLP tasks and algorithms focusing on this topic.

This dissertation explores ways to model and incorporate key narrative elements for understanding and generating stories. We design new controllable generation frameworks to incorporate user specifications in the form of narrative elements into the generation phase. We additionally introduce tasks and resources specifically aimed towards advancing narrative understanding.

1.1 Dissertation Outline

One important narrative element is the *plot*. In Chapter 2, we develop an interactive approach where a user provides the model with plot points in the form of *cue phrases* describing actions, entities, or topic words. Cue phrases help the user inform the system of what they want to happen next, thereby building the plot dynamically. To this end, we propose content-inducing approaches built on top of Transformer networks (Vaswani et al., 2017) to fuse the general decoding and auxiliary cue phrase information through a hierarchical attention mechanism. This human-computer collaboration is a step towards building an AI agent that can communicate with and assist humans (e.g., writers,

learners with disabilities, and children) in a real-world situation and in an effective, scalable way.

Apart from the plot, another key element that makes stories relatable and engaging to readers is the *evolution of emotions*. Cognitive scientists have pinpointed the central role of emotions in storytelling and human communication through conversation. Therefore, in Chapter 3, we propose an emotion-aware story generation task to take into account the emotional trajectory of the protagonist as a narrative element. Inspired by Prince's change-of-state formalization (Prince, 1973), we define an emotion arc describing the protagonist's starting, body, and ending emotional states throughout the story. To incorporate emotional states as controllable attributes, we designed two Emotion-Consistency rewards to enforce the desired emotion arcs using Reinforcement Learning while preserving content quality. This ensures the model to have a high-level understanding of the protagonist's emotional state through the course of the story while generating coherent stories.

Characters are another principal narrative elements. Understanding and critically analyzing *characters* is an important element of understanding a literary piece. Human readers build a mental model of characters, understand what they look like, their role in the literary piece, and assess their psychology, motivations, and consequences of their behavior. However, building such a deep understanding of characters in narratives is hard for machine reading systems. The ultimate goal of character comprehension is to equip machines with these human abilities, which has practical applications in developing story generation (Riedl and Young, 2010b) and chatbot systems (Mairesse and Walker, 2007; Zhang et al., 2018). To bridge the gap between human and machine character-centric narrative understanding, in Chapter 4, we introduce a new dataset containing about 9.5k literature summaries paired with descriptions of characters that appear in them. Using this dataset, we propose tasks to investigate neural models' ability to understand the narrative from the perspective of characters. More specifically, we introduce a novel textual entailment task linking an anonymized character description to the literature summaries. We also introduce character description generation, a more

challenging task requiring the model to analyze the literary piece from the character perspective and then generate a coherent description about them. Performing human assessments on the model outputs shows that there is still much room for improvement on these tasks.

In Chapter 5, we expand fictional characters to a broader set of *entities* as narrative elements. We focus on a real-world application of generating biographical stories about entities grounded in world knowledge. To this end, we propose a grounded keys-to-text generation framework, where the user controls the information content using *factual* and *topical* guiding keys. We developed ranking-based models to fetch relevant information from external knowledge sources and generate the story. To address this task, and assess our model performance, we also present a new dataset, called EntDeGen, and a new evaluation metric of factuality, called MAFE. While proposed ranking strategies proved effective in providing relevant information, human evaluations showed that models still struggle to achieve human-level factuality when generating longer text.

We conclude this thesis with summary of contributions and discussion on future challenges and directions.

Chapter 2

Interactive Story Generation Via Content-Inducing

This chapter discusses work originally published in Brahman et al. (2020).

In this chapter, we explored interactive story generation where we model the *plot* of the story via user-provided *cue phrases* as a narrative element. We present two content-inducing approaches based on the Transformer Network (Vaswani et al., 2017) for interactively incorporating external knowledge when automatically generating stories. Here, our external knowledge is in the form of cue phrases provided by the user, but can readily be replaced with knowledge accessible through other means.[1]

2.1 Introduction

Automatic story generation requires composing a coherent and fluent passage of text about a sequence of events. Prior studies on story generation mostly focused on symbolic planning (Lebowitz, 1987; Pérez y Pérez and Sharples, 2001; Porteous and Cavazza, 2009; Riedl and Young, 2010a) or case-based reasoning (Gervás et al., 2005) that heavily relied on manual knowledge engineering.

Recent state-of-the-art methods for story generation (Martin et al., 2018; Clark et al.,

[1]For example, the user-provided cues can be replaced by the outputs of an automatic planner. Our models are flexible enough to work in other setups.

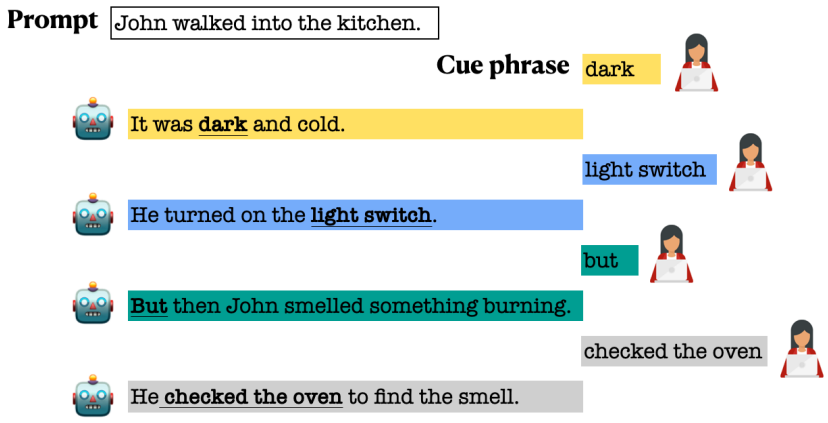


Figure 2.1: Interactive story generation: the user inputs the first sentence of the story (prompt), and provides guiding cue phrases as the system generates the story one sentence at a time.

2018a) are based on sequence-to-sequence models (Sutskever et al., 2014) that generate a story in one go. In this setting, the user has little control over the generated story.

On the other hand, when humans write, they incrementally edit and refine the text they produce. Motivated by this, rather than generating the entire story at once, we explore the problem of interactive story generation. In this setup, a user can provide the model mid-level sentence abstractions in the form of *cue phrases* as the story is being generated. Cue phrases enable the user to inform the system of what they want to happen next in the story and have more control over what is being generated. To achieve our goal, this chapter primarily focuses on approaches for smoothly and effectively incorporating user-provided cues. The schematic in Figure 2.1 illustrates this scenario: the system generates the story one sentence at a time, and the user guides the content of the next sentence using cue phrases. We note that the generated sentences need to fit the context, and also be semantically related to the provided cue phrase.

A fundamental advantage of using this framework as opposed to a fully automated one is that it can provide an interactive interface for human users to incrementally supervise the generation by giving signals to the model throughout the story generation process. This human-computer collaboration can result in generating richer and personalized stories. In particular, this field of research can be used in addressing the literacy

needs of learners with disabilities and enabling children to explore creative writing at an early age by crafting their own stories.

In this chapter, we present two content-inducing approaches based on the Transformer Network (Vaswani et al., 2017) for interactively incorporating external knowledge when automatically generating stories. Here, our external knowledge is in the form of cue phrases provided by the user to enable interaction, but can readily be replaced with knowledge accessible through other means[2]. Specifically, our models fuse information from the story context and cue phrases through a hierarchical attention mechanism. The first approach, *Cued Writer*, employs two independent encoders (for incorporating context and cue phrases) and an additional attention component to capture the semantic agreement between the cue phrase and output sentence. The second approach, *Relevance Cued Writer*, additionally measures the relatedness between the context and cue phrase through a context-cue multi-head unit. In both cases, we introduce different attention units in a single end-to-end neural network.

2.2 Interactive Story Generation

We design models to generate a story one sentence at a time. Given the generated context so far (as a sequence of tokens) $X = \{x_1, ..., x_T\}$, and the cue phrase for the next sentence $c = \{c_1, ..., c_K\}$, our models generate the tokens of the next sentence of the story $Y = \{y_1, ..., y_M\}$. We train the models by minimizing the cross-entropy loss:

$$L_\theta = -\sum_{i=1}^{M} \log P(y_i|X, c, \theta) \tag{2.1}$$

Here, θ refers to model parameters. Note that when generating the n-th sentence, the model takes the first $n-1$ sentences in the story as the context along with the user-provided cue phrase.

In the rest of this section, we describe our two novel content-inducing approaches for

[2]For example, the user-provided cues can be replaced by the outputs of an automatic planner. Our models are flexible enough to work in other setups.

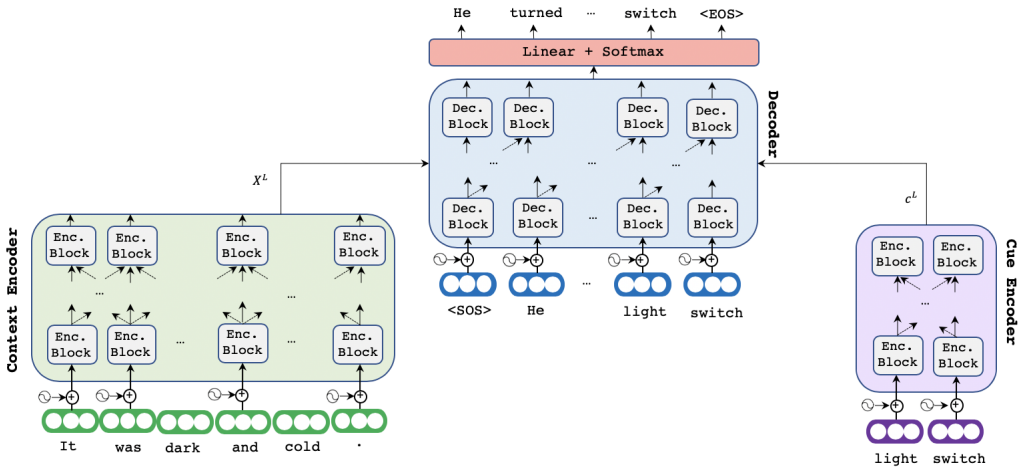


Figure 2.2: Overall model architecture for *Cued Writer* and *Relevance Cued Writer*.

addressing the interactive story generation task: the *Cued Writer*, and the *Relevance Cued Writer*. These models share an overall encoder-decoder based architecture shown in Figure 2.2. They adopt a dual encoding approach where two separate but architecturally similar encoders are used for encoding the context (Context Encoder represented in the green box) and the cue phrase (Cue Encoder represented in the purple box). Both these encoders advise the Decoder (represented in the blue box), which in turn generates the next sentence. The two proposed models use the same encoding mechanism (described in § 2.2.1) and differ only in their decoders (described in § 2.2.2).

2.2.1 Encoder

Our models use the Transformer encoder introduced in (Vaswani et al., 2017). Here, we provide a generic description of the encoder architecture followed by the inputs to this architecture for the Context and Cue Encoders in our models.

Each encoder layer l contains architecturally identical Encoder Blocks, referred to as EncBlock (with unique trainable parameters). Figure 2.3(a) shows an Encoder Block which consists of a Multi-Head attention and an FFN that applies the following operations:

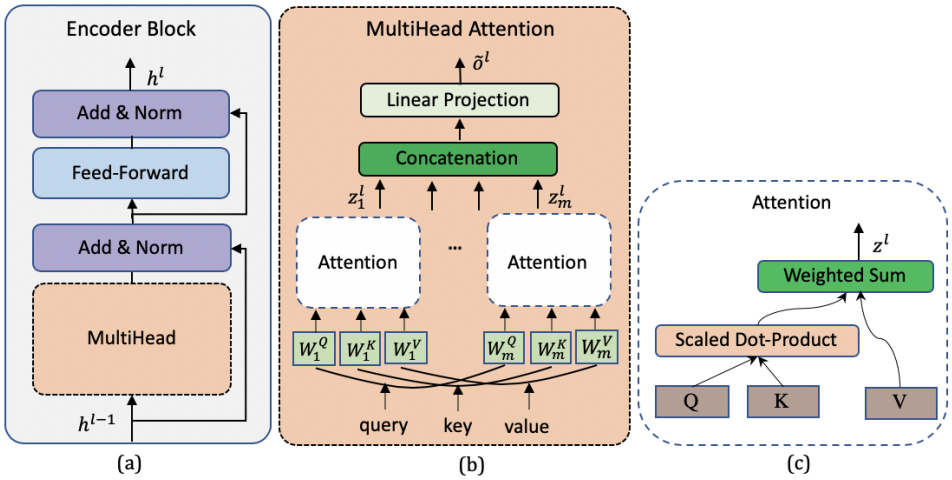


Figure 2.3: (a) Encoder Block consists of MultiHead and FFN. **(b)** MultiHead Attention. **(c)** Attention Module.

$$\tilde{o}^l = \text{MULTIHEAD}(h^{l-1}) \tag{2.2a}$$

$$o^l = \text{LAYERNORM}(\tilde{o}^l + h^{l-1}) \tag{2.2b}$$

$$\tilde{h}^l = \text{FFN}(o^l) \tag{2.2c}$$

$$h^l = \text{LAYERNORM}(\tilde{h}^l + o^l) \tag{2.2d}$$

Where MULTIHEAD represents Multi-Head Attention (described below), FFN is a feed-forward neural network with ReLU activation (LeCun et al., 2015), and LAYERNORM is a layer normalization (Ba et al., 2016).

Multi-Head Attention The multi-head attention, shown in Figure 2.3(b), is similar to that used in Vaswani et al. (2017). It is made of multiple Attention heads, shown in Figure 2.3(c). The Attention head has three types of inputs: the query sequence, $Q \in R^{n_q \times d_k}$, the key sequence, $K \in R^{n_k \times d_k}$, and the value sequence, $V \in R^{n_v \times d_k}$. The attention module takes each token in the query sequence and attends to tokens in the key sequence using a scaled dot product. The score for each token in the key sequence

is then multiplied by the corresponding value vector to form a weighted sum:

$$\mathrm{ATTN}(\mathrm{Q}, \mathrm{K}, \mathrm{V}) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.3}$$

For each head, all Q, K, and V are passed through a head-specific projection prior to the attention being computed. The output of a single head is:

$$H_i = \mathrm{ATTN}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.4}$$

Where Ws are head-specific projections. Attention heads H_i are then concatenated:

$$\mathrm{MULTIH}(\mathrm{Q}, \mathrm{K}, \mathrm{V}) = [H_i; ...; H_m]W^O \tag{2.5}$$

W^O is an output projection. In the encoder, all query, key, and value come from the previous layer and thus:

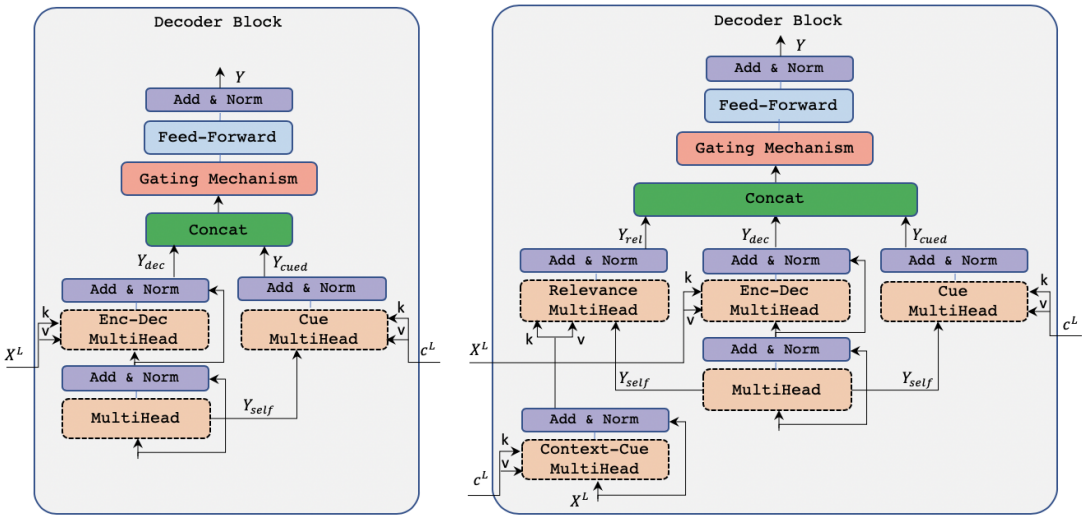
$$\mathrm{MULTIHEAD}(h^{l-1}) = \mathrm{MULTIH}(h^{l-1}, h^{l-1}, h^{l-1}) \tag{2.6}$$

Encoder Input The Encoder Blocks described above form the constituent units of the Context and Cue Encoders, which process the context and cue phrase respectively. Each token in the context, x_i, and cue phrase, c_i, is assigned two kinds of embeddings: *token embeddings* indicating the meaning and *position embeddings* indicating the position of each token within the sequence. These two are summed to obtain individual input vectors, X^0, and c^0, which are then fed to the first layer of Context and Cue encoders, respectively. Thereafter, new representations are constructed through layers of encoder blocks:

$$X^{l+1} = \mathrm{ENCBLOCK}(X^l, X^l, X^l) \tag{2.7a}$$

$$c^{l+1} = \mathrm{ENCBLOCK}(c^l, c^l, c^l) \tag{2.7b}$$

where $l \in [0, L-1]$ denotes different layers.



(a) Cued Writer **(b)** Rel. Cued Writer

Figure 2.4: Decoder architectures. X^L and c^L are the outputs of the top-layers of the Context and Cue encoders respectively, and K and V are the corresponding keys and values.

2.2.2 Content-Inducing Decoders

Cued Writer The main intuition behind our first model, *Cued Writer*, is that since cue phrases indicate users' expectations of what they want to see in the next sentence of the story, they should be used by the model *at the time of generation*, i.e., in the decoder. Below, we describe the *Cued Writer* decoder.

After processing the two types of inputs in the Context and Cue Encoders, the model includes their final encoded representations (X^L and c^L) in the decoder. The decoder consists of L layers with architecturally identical Decoder Blocks. Each Decoder Block contains `Enc-Dec MultiHead` and the `Cue MultiHead` units (see Figure 2.4(a)), which let the decoder in focus on the relevant parts of the context and the cue phrase, respectively.

Given Y^0 as the word-level embedding for the output, our Decoder Block is formulated as:

$$Y_{self}^{l+1} = \text{MULTIH}(Y^l, Y^l, Y^l) \tag{2.8a}$$

$$Y_{dec}^{l+1} = \text{MULTIH}(Y_{self}^{l+1}, X^L, X^L) \tag{2.8b}$$

$$Y_{cued}^{l+1} = \text{MULTIH}(Y_{self}^{l+1}, c^L, c^L) \tag{2.8c}$$

Eqn. 2.8a is standard self-attention, which measures the intra-sentence agreement for the output sentence and corresponds to the `MultiHead` unit shown in Figure 2.4(a). Eqn. 2.8b, describing the `Enc-Dec MultiHead` unit, measures the agreement between context and output sentence, where queries come from the decoder Multi-Head unit (Y_{self}), and the keys and values come from the top layer of the context encoder (X^L). Similarly, Eqn. 2.8c captures the agreement between output sentence and cue phrase through `Cue MultiHead` unit.

Lastly, we adapt a gating mechanism (Sriram et al., 2017) to integrate the semantic representations from both Y_{dec} and Y_{cued} and pass the resulting output to FFN function:

$$g^{l+1} = \sigma(W_1[Y_{dec}^{l+1}; Y_{cued}^{l+1}]) \tag{2.9a}$$

$$Y_{int}^{l+1} = W_2(g^{l+1} \circ [Y_{dec}^{l+1}; Y_{cued}^{l+1}]) \tag{2.9b}$$

$$Y^{l+1} = \text{FFN}(Y_{int}^{l+1}) \tag{2.9c}$$

the representation from Y_{dec} and Y_{cued} are concatenated to learn gates, g. The gated hidden layers are combined by concatenation and followed by a linear projection with the weight matrix W_2.

Relevance Cued Writer The decoder of *Cued Writer* described above captures the relatedness of the context and the cue phrase to the generated sentence but does not study the relatedness or relevance of the cue phrase to the context. We incorporate this relevance in the decoder of our next model, *Relevance Cued Writer*. Its Decoder Block (shown in Figure 2.4(b)) is similar to that of *Cued Writer* except for two additional units: the Context-Cue and Relevance MultiHead units. The intuition behind the `Context-Cue`

`MultiHead` unit (Eqn. 2.10a) is to characterize the relevance between the context and the cue phrase, so as to highlight the effect of words in the cue phrase that are more relevant to the context thereby promoting topicality and fluency. This relevance is then provided to the decoder using the `Relevance MultiHead` unit (Eqn. 2.10b):

$$X_{rel}^{l+1} = \text{MULTIH}(X^L, c^L, c^L) \tag{2.10a}$$

$$Y_{rel}^{l+1} = \text{MULTIH}(Y_{self}^{l+1}, X_{rel}^{l+1}, X_{rel}^{l+1}) \tag{2.10b}$$

We fuse the information from all three sources using a gating mechanism and pass the result to FFN:

$$g^{l+1} = \sigma(W_1[Y_{dec}^{l+1}; Y_{cued}^{l+1}; Y_{rel}^{l+1}]) \tag{2.11a}$$

$$Y_{int}^{l+1} = W_2(g^{l+1} \circ [Y_{dec}^{l+1}; Y_{cued}^{l+1}; Y_{rel}^{l+1}]) \tag{2.11b}$$

$$Y^{l+1} = \text{FFN}(Y_{int}^{l+1}) \tag{2.11c}$$

Finally, for both models, a linear transformation and a Softmax function (shown in Figure 2.2) is applied to convert the output produced by the stack of decoders to predicted next-token probabilities:

$$P(y_i|y_{<i}, X, c, \theta) = \text{softmax}(Y_i^L W_y) \tag{2.12}$$

where $P(y_i|y_{<i}, X, c, \theta)$ is the likelihood of generating y_i given the preceding text ($y_{<i}$), context and cue, and W_y is the token embedding matrix.

2.3 Empirical Evaluation

2.3.1 Dataset

We used the ROCStories corpus (Mostafazadeh et al., 2016) for our experiments. It contains $98,161$ five-sentence long stories. This corpus has a rich set of causal/temporal sequence of everyday events. We held out 10% of stories for each validation and test set.

2.3.2 Baselines

SEQ2SEQ Our first baseline is based on a LSTM sentence-to-sentence generator with attention (Bahdanau et al., 2015). We concatenate context and cue phrase with a delimiter token (<$>) before passing it to the encoder.

DYNAMIC This is the Dynamic model proposed by Yao et al. (2019) modified to work in our setting. For a fair comparison, instead of generating a plan, we provide the model with cue phrases and generate the story one sentence at a time.

STATIC The STATIC model (Yao et al., 2019) gets all cue phrases at once to generate the entire story. By design, it has additional access to all, including future, cue phrases.

VANILLA To verify the effectiveness of our content-inducing approaches, we use Vanilla Transformer as a baseline and concatenate context and cue phrase using a delimiter token.

2.3.3 Training Details

Cue-phrases for Training and Automatic Evaluation: For training all models, we need cue phrases, which are, in principle, to be entered by a user. However, to scale model training, we automatically extracted cue phrases from the target sentences in the training set using the RAKE algorithm (Rose et al., 2010). Our automatic evaluations were done on a large-scale, and so we followed a similar approach for extracting cue-phrases.

Cue-phrases for Human Evaluation: In the interest of evaluating the interactive nature of our models, cue-phrases were provided manually during human evaluations.

2.3.4 Automatic Evaluation

Following previous credible works (Martin et al., 2018; Fan et al., 2018b), we compare methods using Perplexity and BLEU (Papineni et al., 2002) on the test set. From Table 2.1, we can see that both our models outperform DYNAMIC and STATIC by large margins on perplexity and BLEU scores. The proposed models are also superior to the SEQ2SEQ and VANILLA baseline on both measures. Comparing the last two rows, we see an additive

Models	PPL (↓)	B-1 (↑)	B-2 (↑)	B-3 (↑)	GM (↑)	Rep-4 (↓)
DYNAMIC (Yao et al., 2019)	29.49	30.05	9.16	4.59	0.73	44.36
STATIC (Yao et al., 2019)	20.81	33.25	9.64	4.77	0.75	26.26
SEQ2SEQ	20.97	33.91	10.01	3.09	0.82	33.23
VANILLA	15.78	40.30	16.09	7.19	0.89	20.87
Cued Writer	14.80	41.50	16.72	7.25	0.92	**15.08**
Rel. Cued Writer	**14.66**	**42.65**	**17.33**	**7.59**	**0.94**	16.23

Table 2.1: Automatic evaluation results for Perplexity (PPL), BLEU-1 (B-1), BLEU-2 (B-2), BLEU-3 (B-3), Greedy Matching (GM), and Repetition-4 (Rep-4). Our models outperform all baselines across all metrics ($p < 0.05$).

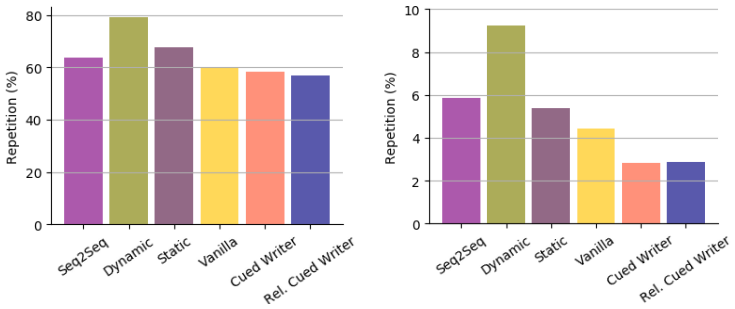


Figure 2.5: Inter-story (left) and Intra-story (right) repetition scores. The proposed models have better scores.

gain from modeling the relevance in *Rel. Cued Writer*.

To evaluate how well the story generation model incorporates the cues, we use an embedding-based score (GM) that measure the relatedness of the generated story with cues by greedily matching them with each token in a story based on the cosine similarity of their word embeddings. We see from the 5th column in Table 2.1 that our models generate stories that are more related to the cue phrases.

Previous works have shown that neural generation models suffer from repetition issue; and so we additionally evaluate the models using repetition-4 which measures the percentage of generated stories that repeat at least one 4-gram (Shao et al., 2019) and inter- and intra-story repetition scores (Yao et al., 2019). A lower value is better for these scores. The result of repetition-4 is reported in the last column of Table 2.1. The proposed models significantly outperform all baselines, and among the two *Cued Writer* is better. Inter and intra repetition scores are depicted in Figure 2.5. Our two proposed models are almost comparable on these metrics but they show a general

superior performance compared to all baselines. In particular, *Rel. Cued Writer* achieves a significant performance increase of 16% and 46% on these scores over the stronger model of Yao et al. (2019).[3]

2.3.5 Human Evaluation

Automatic metrics cannot evaluate all aspects of open-ended text generation, and so we also conduct several human evaluations.

Story-level Evaluation In this experiment, we make pairwise comparisons, where both models are provided the same prompts, and sets of cue phrases.[4] 3 judges evaluated 100 pairs of stories. Figure 2.6(a) shows the percentage of preference for our stronger model, *Rel. Cued Writer*, over the baselines. Judges prefer our model over all other baselines. Also, judges preferred *Rel. Cued Writer* over *Cued Writer*, which demonstrates the effectiveness of the additional Context-Cue and Relevance Multi-Head units.

Sentence-level Evaluation We also performed a more fine-grained evaluation of the models by evaluating generated sentences while the model is generating a story. Specifically, we provide an (incomplete) story passage and a cue phrase to the two models to generate the next sentence. We asked human judges to identify which of the two sentences is better based on their fluency and semantic relevance to (1) the input (incomplete) story and (2) the cue phrase. We did this experiment for a set of 100 randomly selected stories (400 sentences. 3 different judges evaluated each sentence pair. Figure 2.6(b) shows that the *Rel. Cued Writer* model was preferred over SEQ2SEQ and VANILLA in 72% and 64% of the cases, respectively. Comparing the two proposed models, we again see additive gain by modeling Cue-Context relevance. All improvements are statistically significant ($p < 0.05$).

[3] Note that the result of our SEQ2SEQ baseline is not directly comparable with that of Inc-S2S in (Yao et al., 2019), since we included cue phrases as additional input whereas Inc-S2S generate the whole story conditioned on the title.

[4] Cue-phrases were extracted automatically to alleviate any biases, then curated manually.

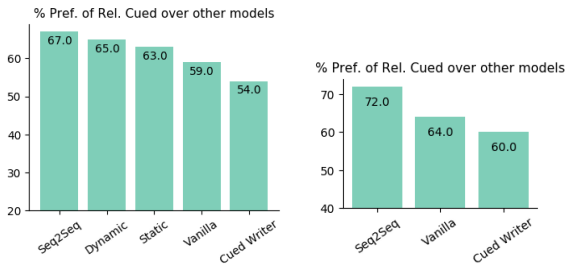


Figure 2.6: Human evaluations on story-level (left) and sentence-level (right).

2.4 Qualitative Results

Table 2.2 presents examples of stories generated by different models for the same prompt and cue phrases. We highlight the [missing] cue phrases, incoherent or unfluent, and repetitive parts of each story. Note that we did not highlight [missing], if the model mentions part of the cue phrase or incorporates it semantically. As we observe, all of the baselines suffered from several issues; however, our novel content inducing approaches generate more causally related sentences, which fit the given prompt and cue phrases more naturally.

Prompt	I was coughing very badly. | **Cue phrases** notice blood - emergency room - waited - okay
DYNAMIC	I noticed that I had a headache. I went to the doctor. The doctor told me that I had to go to **the doctor**. The doctor said I was fine. **[waited]**
STATIC	**She** noticed a huge blood pressure. She went to the doctor. **The doctor told her to stop.** She waited for a while. **[okay]**
SEQ2SEQ	I noticed blood and **noticed blood** in the closet. **Now I have emergency room** for **emergency room**. I found it and I waited until the next day. I feel guilty and now i don't have to do it again. **[okay]**
VANILLA	I noticed blood in the bathroom. I had to go to the emergency room. I waited in the bathroom for a while. Luckily **the doctor was okay.**
Cued Writer	I noticed blood in my mouth. I had to go to the emergency room. I waited there a few hours. I felt okay.
Rel Cued Writer	I noticed blood on my shirt. I went to the emergency room. I waited for the doctor. He came and I was okay.
Prompt	Harry really wanted to see the new Star Wars movie. | **Cue phrases** opening day - ticket - entire movie - really
DYNAMIC	He went to the theater to show up. He bought a ticket. He had a great time. **He had a great time. [opening day] [entire movie]**
STATIC	One day, **she** was in a hurry to get to the theater. she wasn't sure if she would like to go. She was excited to try the new movie. **She was so excited** to see the movie. **[opening day] [ticket]**
SEQ2SEQ	The day **day** was opening day and his family was **opening** the **opening day**. Harry had bought a ticket and the **ticket** wasn't very good. The entire movie was very happy. Harry became **very really** disappointed.
VANILLA	On opening day, Harry was very nervous. He bought a ticket **to** the theater. He bought **Harry** ticket **tickets** to the theater. He really didn't like the movie. **[entire movie]**
Cued Writer	On opening day, he went to the theater . He bought a ticket at the theater. The entire movie was great. He really was excited.
Rel Cued Writer	He decided to watch it on opening day. He got to the theater and got a ticket. He watched the entire movie. He was really excited about it.

Table 2.2: Sample stories generated by different models.

2.5 Related Work

Automatic story generation is a longstanding and challenging problem in AI, with early work dating back to the 1970s based on symbolic planning (Lebowitz, 1987; Pérez y Pérez and Sharples, 2001; Porteous and Cavazza, 2009; Riedl and Young, 2010a) and case-based reasoning using ontologies (Gervás et al., 2005). Li et al. (2013) extended prior works toward learning domain models (via corpus and/or crowd-sourcing) to support open story generation about any topic.

With the advent of deep learning there has been a major shift towards using seq2seq models (Sutskever et al., 2014; Bahdanau et al., 2015) for various text generation tasks, including storytelling (Roemmele, 2016; Jain et al., 2017; Hu et al., 2020). However, these models often fail to ensure coherence in the generated story. To address this problem, Clark et al. (2018a) incorporated entities given their vector representations, which get updated as the story unfolds. Similarly, Liu et al. (2020) proposed a character-centric story generation by learning character embeddings directly from the corpus. Fan et al. (2018b) followed a two-step process to first generate the premise and then condition on that to generate the story. Yu et al. (2020) proposed a multi-pass CVAE to improve wording diversity and content consistency.

Previous work has explored the potential of creative writing with a machine in the loop. Clark et al. (2018c) found that people generally enjoy collaborating with a machine. Traditional methods proposed to write stories collaboratively using a case-based reasoning architecture (Swanson and Gordon, 2012). Recent work (Roemmele and Gordon, 2015) extended this to find relevant suggestions for the next sentence in a story from a large corpus. Unlike us, these approaches explore the value of and tools for interaction rather than designing methods for incorporating user input into the model.

Another line of research decomposes story generation into two steps: story plot planning and plot-to-surface generation. Previous work produces story-plans based on sequences of events (Martin et al., 2018; Tambwekar et al., 2019; Ammanabrolu et al., 2020), critical phrases (Xu et al., 2018) or both events and entities (Fan et al., 2019). Yao

et al. (2019) model the story-plan as a sequence of keywords. They proposed *Static* and *Dynamic* paradigms that generate a story based on these story-plans. Goldfarb-Tarrant et al. (2019) adopted the *static* model proposed in Yao et al. (2019) to supervise story-writing.

2.6 Summary

We explored interactive story generation by modeling the plot of the story via cue phrases as a narrative element. We presented two content-inducing approaches that take user-provided inputs as the story progresses and effectively incorporate them in the generated text. Results show that our methods outperform competitive baselines along various evaluation axes and yield better stories.

Chapter 3

Modeling Emotional Development for Emotion-Aware Storytelling

This chapter discusses work originally published in Brahman and Chaturvedi (2020).

Since emotions and their evolution play a central role in creating a captivating story, in this chapter, we propose to use *the evolution of emotions* as a narrative element for the task of emotion-aware storytelling. We design methods to generate stories that adhere to given story titles and desired *emotion arcs* for the protagonist. Inspired by Prince's change-of-state formalization (Prince, 1973), we define the *emotion arc* as a sequence of three *basic emotions* (*anger, fear, joy, sadness* (Jack et al., 2014), and *neutral*) that describe the starting, body, and ending emotional states of the protagonist. Our models include Emotion Supervision (EmoSup) and two Emotion-Reinforced (EmoRL) models. The EmoRL models use special rewards (Ec-Em and Ec-Clf) designed to regularize the story generation process through reinforcement learning.

3.1 Introduction

Cognitive scientists have pinpointed the central role of emotions in storytelling (Parkinson and Manstead, 1993; Hogan, 2011). Early automatic storytelling systems based on symbolic planning also showed that addressing character emotions for plot construction resulted in more diverse and interesting stories (Theune et al., 2004; Pérez y Pérez, 2007;

Title (input): Raw burger
Emotion arc (input): joy → anger → sadness
Story (output): Tom went to a burger place with his friends. He ordered a burger. When he got it , he noticed that it was raw. Tom yelled at the waiter for it being raw. He was really disappointed.

Figure 3.1: An example story generated by our model for a given title and emotion arc of the protagonist. Story segments are highlighted with the emotions the protagonist (Tom) experiences.

Méndez et al., 2016). However, these studies were rule-based and limited to small-scale data. The advent of deep learning has shifted computational storytelling efforts towards neural methods (Martin et al., 2018; Yao et al., 2019). However, despite the broad recognition of its importance, neural story generation methods have not explored the modeling of emotional trajectory.

In this chapter, we present the first study to take into account the emotional trajectory of the protagonist in neural story generation. Research in cognitive science has shown that while comprehending narratives, readers closely monitor the protagonist's emotional states (Komeda and Kusumi, 2006; Gernsbacher et al., 1992). However, emotions experienced by the protagonist might differ from the general emotions expressed in the story. For example, the general emotion of *"My boss was very angry and decided to fire me."* is *anger*, but the narrator's emotional reaction would be to feel *upset*. At any point in a story, we represent the protagonist's emotions using a set of *basic emotions*. The theory of basic emotions is well-accepted in psychology, but there is little consensus about the precise number of basic emotions. Plutchik (1982) proposed 8 primary emotions, and Ekman (1992) first proposed 7 and then changed to 6 basic emotions. Following recent theories (Jack et al., 2014; Gu et al., 2016), we choose *anger, fear, joy*, and *sadness*, to describe the protagonist's emotions. We additionally include *neutral* to account for cases with no strong emotions. We refer to these 5 emotions as *basic emotions*.

Moreover, emotions evolve within a narrative. For modeling the evolving emotions of the protagonist, we define an *emotion arc* for a story. Our definition is inspired by Prince's change-of-state formalization (Prince, 1973), which asserts that stories are about change. According to this theory, a story has three components: a starting state; an

ending state; and events that translate the starting into the ending state. Motivated by this, we define the *emotion arc* as a sequence of three *basic emotions* that describe the starting, body, and ending emotional states of the protagonist.

Given a story title and the emotion arc of the protagonist as inputs, our goal is to generate a story about the title that adheres to the given emotion arc. Fig. 3.1 shows an example story generated by our model, where the protagonist's emotion evolves from *joy* to *anger* and then *sadness*.

To address this problem, we present three models based on GPT-2 (Radford et al., 2019) that incorporate the protagonist's emotion arc as a controllable attribute while preserving content quality: an Emotion Supervision (EmoSup), and two Emotion-Reinforced (EmoRL) models based on reinforcement learning. The EmoRL models use two Emotion-Consistency rewards, Ec-Em and Ec-Clf. Ec-Em uses semantically enhanced emotion matching to encourage the model to adhere to the given emotion arc. It infers the protagonist's emotions in the generated stories using *Commonsense Transformers*, COMET (Bosselut et al., 2019). Ec-Clf achieves the same goal using a classifier to infer the protagonist's emotions.

In the absence of a training corpus of stories labeled with the protagonist's emotions, we automatically annotate a large-scale story corpus using COMET. Our automatic and manual evaluations show that our models can not only express the desired emotion arcs but also produce fluent and coherent stories.

3.2 Emotion-Aware Storytelling

We formulate the emotion-aware storytelling task as follows: given a story title as a sequence of tokens $\boldsymbol{t}=\{t_1, t_2, ..., t_m\}$, and an emotion arc for the protagonist as a sequence of *basic emotions* $\boldsymbol{a}=\{e_1, e_2, e_3\}$, the task is to generate a story as a sequence of tokens $\boldsymbol{y}=\{y_1, y_2, ..., y_n\}$ that adheres to the title and emotion arc.

In the rest of the chapter, we first explain how our models can track the protagonist's emotional trajectory (§3.2.1). We then provide an introduction to our base storytelling

model (§3.2.2), which is used as the backbone of our three proposed models (§3.2.3 and §3.2.4).

3.2.1 Tracking Protagonist's Emotions

In this work, we define the *protagonist* as the most frequently occurring character in a narrative (Morrow, 1985). Our two rewards for the EmoRL models (§3.2.4) need to track the protagonist's emotions to guide the generation. For this, we obtain his/her emotions at various stages in the story using one of the following two approaches:

Commonsense Transformer Our Ec-Em reward uses a commonsense knowledge model to reason about the implicit emotional states of the protagonist. We use COMET (Bosselut et al., 2019), a knowledge base construction model trained on ATOMIC *if-then* knowledge triples (Sap et al., 2019). It contains information about everyday events and their causes and effects. Given an event and a relation, COMET can generate commonsense inferences about the relation (e.g,"if X pays Y a compliment, then Y feels flattered"). For tracking emotions, we use relations `xReact` and `oReact` that correspond to emotional reactions to events (more details on this in §3.3.1).

Emotion Classifier Our Ec-Clf reward captures the protagonist's emotions using an emotion classifier. For this, we adapt the pre-trained BERT$_{large}$ for multi-label classification over 5 *basic emotions*: *anger, fear, joy, sadness,* and *neutral*. Following (Devlin et al., 2019), we use a fully-connected layer over the final hidden representation corresponding to the special classification token (`[CLS]`). We train this classifier in two steps. First, we train this classifier on a human-annotated dataset for emotion identification in tweets (Mohammad et al., 2018), we then further fine-tune the classifier on story training data that is automatically annotated with the protagonist's emotions using the pipeline described in §3.3.1. To evaluate the classifier, we collect manual annotations for the protagonist's emotions on Amazon Mechanical Turk for a subset of 50 randomly selected stories (250 sentences) from our story corpus. Each sentence was annotated by 3 judges. Workers agreed with our emotion classifier 70% of the time (random agreement would be 20%). See Appendix A.2.2 for more details about these annotations.

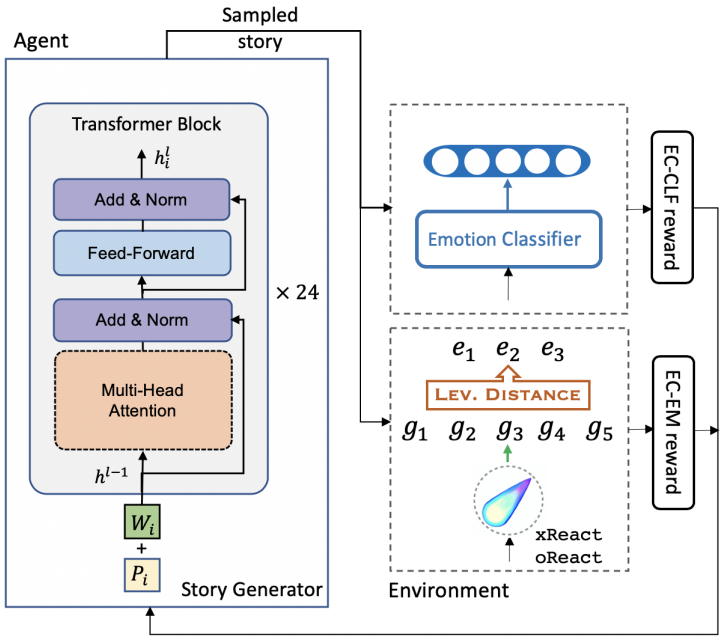


Figure 3.2: Transformer block architecture (left) and emotion-reinforced storytelling framework (right)

3.2.2 Transformer-based Storytelling Model

Our models are built upon a base storytelling model that can generate a story consistent with a given prompt (e.g., title). We choose GPT-2 (medium) (Radford et al., 2019) because our initial experiments demonstrated that it outperforms other state-of-the-art story generation models, in general. GPT-2 uses multiple Transformer blocks of multi-head self-attention and fully connected layers (described in chapter 2 and shown in the left box in Fig. 3.2). Since it was trained on a broad range of domains, we fine-tune it on a dataset of stories by minimizing the conditional log-likelihood:

$$
\begin{aligned}
\mathcal{L}_{ML} &= -\sum_{i=m}^{m+n} \log p(y_i|y_{<i}, \boldsymbol{t}) \\
p(y_i|y_{<i}, \boldsymbol{t}) &= \text{softmax}(h_i^L W^T) \\
h_i^l &= \text{block}(h_{<i}^{l-1}), l \in [1, L] \\
h_i^0 &= W_i + P_i
\end{aligned}
\tag{3.1}
$$

where m and n denote the number of tokens in the title and story respectively. h_i^l is the

l-th layer's output at the i-th position computed through transformer block with the masked multi-head self attention mechanism, and h_i^0 is a summation of *token embedding* W_i and *position embedding* P_i for the i-th token. $y_{<i}$ indicates left context.

3.2.3 Emotion Supervision (EmoSup) Model

The underlying idea behind our Emotion Supervision (EmoSup) model is to provide the emotion arc as an additional input similar to conditional training (Fan et al., 2018a; Kikuchi et al., 2016). Specifically, each title has the corresponding emotion arc prepended at the beginning, separated by a delimiter token (<$>). This way, emotion arcs receive special treatment (Kobus et al., 2017), since they are propagated to all of the story and the model learns to maximize $p(y_i|y_{<i}, \boldsymbol{t}, \boldsymbol{a})$.

3.2.4 Emotion-Reinforced (EmoRL) Models

The emotion arc guides the generation in EmoSup as an initial input. However, we want to continually supervise the model *during* the generation process. This motivates us to use a reinforcement learning framework. Here, we propose two Emotion Consistency rewards, Ec-Em and Ec-Clf, which optimize adherence to the desired emotion arc.

Ec-Em Reward This reward quantifies the alignment of the emotion arc of the generated story to the desired arc using the commonsense knowledge model, COMET. For an N-sentence-long generated story, we use COMET to obtain the protagonist's emotional reaction for each sentence, resulting in a sequence of emotion-phrases $\boldsymbol{a}^g$=$\{g_1, g_2, ..., g_N\}$[1]. We then define the reward as a modified Levenshtein distance (Levenshtein, 1966) between the generated reactions $\boldsymbol{a}^g$ and the desired emotion arc $\boldsymbol{a}^*$=$\{e_1, e_2, e_3\}$. This modification allows only two operations: (1) Deletion of an emotion-phrase (in $\boldsymbol{a}^g$), and (2) Replacement of an emotion-phrase with a *basic emotion* at a cost proportional to semantic similarity between the two (e.g., *happy to help* and *joy*). Semantic similarities are computed using cosine similarity between the averaged GloVe embeddings (Pennington

[1] N=5 for our dataset. Also, COMET's outputs, g_is, are phrases representing emotional reactions. Details on obtaining emotional reactions during training are provided in Appendix A.1.1.

et al., 2014). The reward is defined as:

$$r_{em} = \text{lev}(\boldsymbol{a}^g, \boldsymbol{a}^*) \tag{3.2}$$

where lev denotes the modified Levenshtein distance. We refer to the model that uses this reward as **RL-EM**.

EC-CLF Reward This reward infers the protagonist's emotions in a given text using our emotion classifier (§3.2.1). We first divide the generated story into segments: beginning, body, and ending[2]. Then, for each segment, we use the classifier to obtain the probability of the desired emotion. The reward is defined as the probabilities of the desired emotions averaged across the segments:

$$r_{clf} = \frac{1}{k}\sum_{j=1}^{k} p_{\text{clf}}(e_j^*|\boldsymbol{x}_j) \tag{3.3}$$

where k is the number of tokens in the emotion arc (here, k=3), and e_j^* denotes the desired emotion for j-th segment $\boldsymbol{x}_j$. We refer to the model that uses this reward as **RL-CLF**.

Policy Gradient For training, we use the REINFORCE algorithm (Williams, 1992) to learn a generation policy p_θ of the storytelling model with parameters θ. Here, the model generates a sample story $\boldsymbol{y}^s$ from the model's output distribution, and the goal is to minimize the negative expected reward, which is approximated by:

$$\mathcal{L}_{RL} = -(r(\boldsymbol{y}^s) - r(\hat{\boldsymbol{y}}))\sum_{i=k+m}^{k+m+n} \log p_\theta(y_i^s|y_{<i}^s) \tag{3.4}$$

We follow the self-critical training approach (Rennie et al., 2017), and take the reward of the greedily decoded story $\hat{\boldsymbol{y}}$ as the baseline reward ($r(\hat{\boldsymbol{y}})$). This ensures that with better exploration, the model learns to generate stories $\boldsymbol{y^s}$ with higher rewards than the baseline $\hat{\boldsymbol{y}}$. However, optimizing only with the RL loss mentioned above using the emotion-consistency rewards may increase the expected rewards, but at the cost of fluency and readability of the generated story. Therefore, we optimize the following

[2]We generate 5-sentence long stories similar to our training corpus and segment them into the beginning, body, and ending in 1:3:1 ratio.

mixed loss (Paulus et al., 2018):

$$\mathcal{L}_{mixed} = \gamma \mathcal{L}_{RL} + (1 - \gamma)\mathcal{L}_{ML} \tag{3.5}$$

where γ is a hyper-parameter balancing the two loss functions. Our emotion-reinforced storytelling framework is depicted in Fig. 3.2.

3.3 Experimental Setup

3.3.1 Dataset and Annotation Pipeline

We use the *ROCStories* corpus (Mostafazadeh et al., 2016) for our experiments. It contains $98,162$ five-sentence stories, designed to have a clear beginning and ending, thus making it a good choice for our emotion-aware storytelling task. We held out 10% of the stories each for validation and test sets, respectively.

For training our models, we need stories annotated with emotion arcs of the protagonists. We annotated the stories in our dataset automatically using the multi-step annotation pipeline shown in Fig. 3.3. In step 1, we identify all characters and their mentions in a story using coreference resolution. In step 2, we identify the character with the most mentions as the *protagonist* (e.g., 'Iris' who is mentioned in 4 sentences). Then, in step 3, in each sentence of the story, we identify the protagonist's role as *Agent* or *Other* using its dependency parse. The protagonist is an *Agent* if he/she is the subject of the main verb in the sentence and *Other* otherwise (e.g., Iris's role is *Other* in sentence 4 and *Agent* in

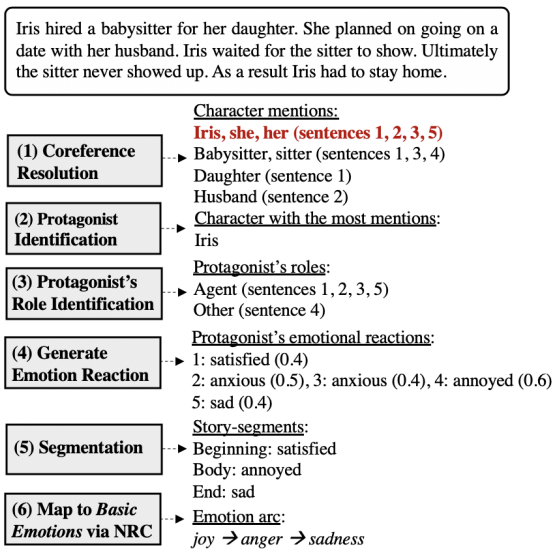


Figure 3.3: Annotation pipeline for *emotion arc*.

all other sentences). Next, in step 4, we obtain the emotional reaction of the protagonist in each sentence using COMET. Given a context, c, and relation type, r, COMET can yield the emotional reaction of the *Agent* (r=xReact) and *Others* (r=oReact). Depending on the protagonist's role in the sentence, we use the appropriate relation to get his/her emotional reaction, g, and COMET's confidence in the prediction, φ_g. In sentences without an explicit mention of the protagonist, his/her role is assigned as *Other*, and we use oReact since the event in that sentence will affect all characters of the story, including the protagonist (e.g., sentence 4 in Fig. 3.3).

Step 4 gives the protagonist's emotions for each sentence of the story, but the emotion arc has to represent them for the three segments: beginning, body, and end. The stories in our corpus are 5-sentence long, and following previous work on this corpus (Chaturvedi et al., 2017b), we segment them in 1:3:1 ratio. For the protagonist's emotion in the body (middle 3 sentences), we take the emotion of the sentence in which COMET was most confident (e.g., 'annoyed' for the body of the running example in Step 5).

Note that since COMET's outputs, gs, are open-ended emotion-phrases, in step 6, we need to map these phrases to one of the 5 *basic emotions* using NRC Affect Intensity Lexicon (Mohammad, 2018). The lexicon is a list of words with their real-valued intensities for 4 non-*neutral* basic emotions. We represent the likelihood of g getting mapped to each of the basic emotions, e, as $score_e(\text{e})$. For mapping, we first tokenize, lemmatize, and filter stop words from g. Then we find exact matches of g's tokens to words in the lexicon (along with the match-intensities). For each match, we increase $score_g(\text{e})$ by the match-intensities. Finally, g is mapped to the basic emotion with the maximum score. An emotion-phrase with no matching tokens is mapped to *neutral*.

3.3.2 Implementation Details

We follow the training and inference settings of medium-size GPT-2 as in Radford et al. (2019) (for completion, we provide full details in Appendix A.1.2). Our models are implemented with the Texar toolkit Hu et al. (2019).

3.3.3 Evaluation Measures

Automatic We adopt several automatic measures to evaluate the generated stories both on content quality and emotion faithfulness.

For evaluating the content quality, we use the following measures: (1) **Perplexity** as an indicator of fluency. A smaller value is better[3]. (2) **BLEU**, which is based on n-gram overlaps (Papineni et al., 2002). Following Guan et al. (2020), since BLEU scores become extremely low for large n, we used n=1, 2. (3) **Distinct-n** (with n=1, 2, 3) measure the percentage of unique n-grams (Li et al., 2016). A high ratio indicates a high level of lexical diversity. (4) **Repetition-4** is the percentage of generated stories that repeat at least one 4-gram (Shao et al., 2019). A high value indicates redundancy in the generated text.

For evaluating the emotion faithfulness of a generated story, we adapt lexical measures (1) **Seg-word** and (2) **Arc-word** (Song et al., 2019). Given a desired emotion arc for a story, Seg-word is the percentage of the story's segments that contain emotion words corresponding to desired emotion. Correspondingly, Arc-word for a story is a binary score indicating if *all* of its segments contain emotion words corresponding to the desired emotions. We also define (3) **Seg-acc** and (4) **Arc-acc** for a generated story. Seg-acc for the story is the fraction of generated segments for which the emotion (as determined by the emotion classifier) exactly matches the desired emotion. Similarly, Arc-acc for a story is a binary score indicating if its emotion arc (as determined by the emotion classifier) exactly matches the desired emotion arc. We also use the reward functions, (5) **Ec-Clf** and (6) **Ec-Em**, to score a generated story. For all these measures, we report averaged scores across all stories generated by a model.

We also conduct a manual evaluation of generated stories using Amazon Mechanical Turk. Workers are asked to evaluate pair of stories on a 0-3 scale (3 being very good) from two different perspectives: (1) **content quality** to indicate whether a story is fluent, logically coherent, and on-topic (related to the given title), and (2) **emotion faithfulness**

[3]For comparison, we compute *word-level* perplexity for GPT-2 based models. That is, we normalize the total negative log probability by the number of *word*-level tokens, not the number of BPE tokens.

Models	PPL (↓)	BLEU-1 (↑)	BLEU-2 (↑)	Dist-1 (↑)	Dist-2 (↑)	Dist-3 (↑)	Repet-4 (↓)
Fusion + Emo	24.02	21.10	2.61	66.18	90.88	96.91	23.30
Plan&Write + Emo	17.43	22.46	3.03	66.32	90.47	95.59	28.61
PPLM-3	–	20.36	2.37	71.37	93.90	98.19	13.36
PPLM-5	–	20.61	2.47	71.47	93.99	98.21	14.02
GPT-2 + FT*	12.16	22.68	3.10	72.93	94.24	98.28	12.10
EmoSup	**11.10**	22.70	3.23	71.44	93.75	98.10	13.94
RL-EM	11.98	22.52	3.15	**73.32**	**94.76**	**98.56**	**10.09**
RL-CLF	11.31	**22.78**	**3.26**	71.16	93.65	98.05	13.34

Table 3.1: Automatic evaluation results for content quality. RL-CLF and RL-EM outperform all baselines for BLEU and diversity/repetition scores respectively ($p < 0.05$). *GPT-2+FT is not provided with emotion arc.

to assess whether it follows the desired emotion arc for the protagonist. Workers were also asked to indicate their **overall preference** by choosing the better story of the two while considering both aspects, or indicate that they are of equal quality. More details about evaluation measures are provided in Appendix A.1.3.

3.3.4 Emotion-Aware Storytelling Results

Baselines We use the following baselines in our experiments: (1) GPT-2+FT, our base GPT-2 model fine-tuned on the ROC stories corpus, for which emotion arcs are not provided as inputs; (2) *Fusion+Emo* and (3) *Plan&Write+Emo*, which are two of the strongest storytelling baselines; and (4) *PPLM* (Dathathri et al., 2020) which can be extended to accept emotion arcs for controlling story generation. *PPLM-3* and *PPLM-5* indicate 3 and 5 iterations respectively.

Automatic Evaluation For automatic evaluation, we used the titles and automatically extracted emotion arcs of the stories in our test set as input.

The evaluation results on content quality are shown in Table 3.1. Interestingly, even though the proposed models only aim to control emotion arc, they outperform GPT-2+FT on perplexity indicating better fluency. Among the proposed models, EmoSup obtains the best perplexity score mainly because that is what its loss function optimizes (as opposed to the mixed loss in EmoRL models). Overall, all of our proposed models outperform all baselines. In particular, RL-CLF has the highest BLEU scores, and RL-EM has the highest diversity and lowest repetition scores. All improvements over baselines

Models	Arc-word	Seg-word	Arc-acc	Seg-acc	EC-CLF	EC-EM
Fusion + Emo	6.32	38.89	29.89	62.59	60.06	73.59
Plan&Write + Emo	5.61	32.98	26.38	60.99	58.13	72.46
PPLM-3	7.11	36.10	23.97	57.01	56.02	73.19
PPLM-5	7.74	37.64	27.30	60.60	59.51	74.43
GPT-2 + FT*	4.46	33.28	17.32	48.69	47.93	69.00
EmoSup	7.33	40.86	31.25	64.26	62.88	74.10
RL-EM	8.85	43.77	33.56	65.13	63.93	**76.59**
RL-CLF	**10.14**	**45.42**	**37.58**	**68.90**	**67.55**	75.87

Table 3.2: Automatic evaluation for emotion faithfulness. For emotion faithfulness, RL-CLF outperforms all baselines ($p < 0.05$). *indicates absence of emotion arc as input.

	Specific Criteria		**Overall Preference**
	Content Quality	Emotion Faithfulness	Better / Worse / Tie (%)
RL-CLF vs. RL-EM	+0.15	+0.20	**52.33** / 38.00 / 9.66
RL-CLF vs. GPT-2+FT	+0.25	+0.76	**60.00** / 22.00 / 18.00
RL-CLF vs. EmoSup	+0.14	+0.28	**50.33** / 34.00 / 15.66
RL-CLF vs. PPLM-5	+0.34	+0.48	**61.00** / 25.66 / 13.33

Table 3.3: Manual evaluation results. For each criteria, we report the average improvements RL-CLF is preferred over other methods ($p < 0.05$).

are statistically significant (approximate randomization (Noreen, 1989), $p < 0.05$).

The evaluation results on emotion faithfulness are shown in Table 3.2. We see that, as expected, all models outperform GPT-2+FT, which is not provided the emotion arcs as inputs. Our proposed models also achieve significant improvements over all baselines (app. randomization, $p < 0.05$). In particular, RL-CLF achieves the best performance on almost all measures.

These results indicate that while all proposed models can control the emotion arc of generated stories, RL-CLF achieves a good balance between both content and emotion quality.

Manual Evaluation Since concerns have been raised about automatic evaluation of language generation, we also conduct a manual evaluation. For this, we randomly sampled titles and emotion arcs of 100 instances from our test set, and generated stories using the models being evaluated. We compared five models, and so overall, there were 500 stories. We conduct pairwise comparisons of generated stories, and each pair was evaluated by 3 judges. Table 3.3 reports the average improvements as well as absolute

Title: fire injuries

joy - sadness - joy	My friends and I went camping this summer. We got in my van and went to the woods. We decided to light a campfire. While driving around , our tire popped and the fire started. We had to call the fire department for help and they were able to put out the fire.
sadness - sadness - joy	The fire department was called to a house in the woods. The house was engulfed in flames. There were two people inside. One person was taken to the hospital by air ambulance. Luckily, the other person was treated for non-life threatening injuries.

Title: dance

fear - joy - joy	Kelly was worried about her dance recital. She had practiced her dance for weeks. She decided to try out for the school's dance team. Kelly was nervous but knew she could do well. She was so excited she gave her best impression!
sadness - joy - joy	I was very depressed. I went to a dance class with a friend of mine. We tried out some different moves. We got stuck dancing for a long time. The next day I tried out some new moves and got a standing ovation.

Table 3.4: For a given title, our model can generate different stories for different emotion arcs. Story segments with corresponding emotions are highlighted.

scores for content quality and emotional faithfulness (evaluated independently) and also the overall preference of the judges. We first compare our two EmoRL models (Row 1). We see that RL-CLF improves over RL-EM on both emotion faithfulness and content quality. Overall, it is judged to be better than RL-EM 52.33% of the time and worse in only 38.00% cases. We then compare the better of the two, RL-CLF, with the uncontrolled GPT-2+FT (Row 2). We see that on average, RL-CLF model is not only better at adhering to the emotion arc by $+0.76$ points but also generates better content (improvement of $+0.25$ points) and its stories are preferred 60% of the times by humans. We observe similar results for comparison with EmoSup and PPLM-5. All improvements are statistically significant (app. randomization, $p < 0.05$).

Case Study Since the proposed models can generate stories conditioned on the protagonist's *emotion arc*, they can be used to unfold a story in diverse situations for a given title. We demonstrate this capability in Table 3.4. It shows two examples where for the same

title, our model (RL-CLF here) can generate stories that follow different emotion arcs for the protagonists. We provide more qualitative examples in Appendix Figure A.4.

3.4 Related Works

Early story generation systems relied on symbolic planning (Lebowitz, 1987; Pérez y Pérez and Sharples, 2001; Porteous and Cavazza, 2009; Riedl and Young, 2010a) or case-based reasoning (Gervás et al., 2005). Although these systems could ensure long-term coherence, they could only operate in predefined domains and required manual engineering. These problems have been somewhat alleviated by recent seq2seq storytelling models (Roemmele, 2016; Jain et al., 2017), some of which are based on intermediate representations (Martin et al., 2018; Xu et al., 2018; Fan et al., 2018c; Yao et al., 2019; Fan et al., 2019).

Recent approaches have also used large-scale language models (LMs) based on Transformers (Vaswani et al., 2017), such as GPT-2 (Radford et al., 2019). Being trained on large amounts of data, these models can generate highly fluent text and find applications in story generation (Qin et al., 2019; Guan et al., 2020) and dialogue systems (Budzianowski and Vulić, 2019; Wolf et al., 2019b). However, they lack the ability to dictate any auxiliary objective for the generated text, such as expressing specific attributes.

To address this, approaches such as conditional training or weighted decoding have been proposed to control different properties of the generated text such as sentiment, tense, speaker style, and length (Kikuchi et al., 2016; Hu et al., 2017; Ghazvininejad et al., 2017; Wang et al., 2017; Fan et al., 2018a). Tambwekar et al. (2019) use reinforcement learning for generating goal-driven story plots, which are sequences of event tuples. Dathathri et al. (2020) propose PPLM, which uses an attribute classifier to steer text generation without further training the LM.

More closely related to our work is generating automatic responses with a specific sentiment or emotion (Zhou et al., 2018; Huang et al., 2018; Zhou and Wang, 2018; Song

et al., 2019). Modeling characters (Bamman et al. (2013a), Bamman et al. (2014a), Vala et al. (2015), Iyyer et al. (2016), Kim and Klinger (2019a), Krishnan and Eisenstein (2015a) *inter alia*) and sentiment trajectory (Chaturvedi et al. (2017b), Chen et al. (2019) *inter alia*) has been shown to be useful for story understanding, in general. However, there are limited works on incorporating characters or sentiment for story generation. Previous work model characters but not sentiment (Clark et al., 2018b; Liu et al., 2020). Weber et al. (2020) is a contemporary and unpublished work that incorporates sentiment while "filling in" a narrative. Peng et al. (2018) and Luo et al. (2019) control the overall sentiment for story ending generation. These works are limited to coarse-grained sentiments and/or only target the ending sentence. Instead, we model the emotional trajectory of the protagonist as the story progresses, which is more central to storytelling than the overall sentiment.

3.5 Summary

In this chapter, we proposed an emotion-aware story generation task for modeling the *evolution of emotions* in stories. To this goal, we designed two emotion-consistency rewards using a commonsense transformer and an emotion classifier. Experiments demonstrated that our approach improved both content quality and emotion faithfulness of the generated stories. This work is a step towards future research directions on planning emotional trajectory while generating stories. Using commonsense inferences about the effect of the events on emotional states of various characters of the story has the potential of generating more coherent, realistic, and engaging stories. In this work, we focused only on the protagonist, but future works can explore modeling motivations, goals, achievements, and emotional trajectory of all characters.

Chapter 4

Character-centric Narrative Understanding

This chapter discusses work originally published in Brahman et al. (2021).

In the previous two chapters, we focused on the problem of automatic story generation while adding controllability and modeling key narrative elements such as *plot* and *emotional development*.

Character(s) are another principal narrative element whose actions, traits, and motivations determine the flow of the plot. When reading a literary piece, readers often make inferences about various characters' roles, personalities, relationships, intents, actions, etc., which are well aligned with the psychological and literary theories about the theory of mind (ToM). While humans can readily draw upon their past experiences to build such a character-centric view of the narrative, *understanding* characters in narratives can be a challenging task for machines. To bridge the gap between human and machine character-centric narrative understanding, in this chapter, we introduce a new dataset and tasks to encourage progress in the field of character-centric NLP models.

4.1 Introduction

Previous works in literary analysis have discussed that the development of the plot and the main character(s) are among the most important components that contribute to a

good piece of fiction (Kennedy and Gioia, 1983; Card, 1999). In particular, *character*(*s*) are central to narratives since their motivations, traits, and actions determine the flow of the plot. Hense, *understanding characters* has been widely recognized as key to understanding stories, by psychology, literary and education 028 research (Bower and Morrow, 1990; Curie, 2009; Dymock, 2007).

In Computational Narratives, prior work has exploited the potential of character-centric natural language understanding (Chambers, 2013; Chaturvedi et al., 2017a; Chu et al., 2018; Zhang et al., 2019). However, these works are limited to only understanding certain aspects of characters and do not do an in-depth and systematic study.

To facilitate character-centric narrative understanding, we present `LiSCU` – a new dataset in English, of literary pieces and their summaries paired with descriptions of characters that appear in them. These descriptions analyze the narrative from the perspective of the character highlighting their salient attributes, their role and contribution to the development of the narrative's plot.

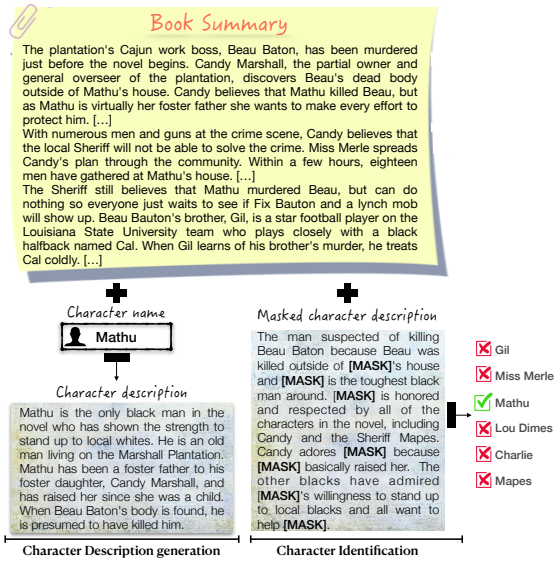


Figure 4.1: An illustration of the proposed dataset and the two tasks: *Character Description Generation* and *Character Identification*.

Using this dataset, we devise two new tasks: (1) a *Character Identification* task to identify the character's name from an anonymized character description given the literature summary; and (2) a *Character Description Generation* task to generate the description for a given character of a literature summary. Our primary task, *Character Description Generation*, is related, but not identical to summarization. There are two main differences. Summarization typically has a one-to-one correspondence between documents and summaries, and focuses on copying (either extractively or abstractively) important content from the documents to create the summaries. On the other hand, char-

acter descriptions are analysis, not merely summaries, of narratives from the character's point of view. They are created by abstracting out the low-level content of the narrative instead of simply identifying and paraphrasing important details. They describe events, roles, relationships, and salient attributes of the character that can be inferred from the narrative and might not be directly stated in the text. In particular, if the narrative describes several events where a character helps the protagonist, the character description will not simply mention all those events, but will instead describe the character as a helpful person (attribute) and a good friend of the protagonist (role). For example, in Figure 4.1, *"Mathu is virtually her foster father."* in summary is expressed as *"Candy adores Mathu because he basically raised her."* in the character description. Thus, the *Character Description Generation* task, provides a unique opportunity for NLP systems to learn to *abstract* and model long-range dependency instead of simply *extracting* information.

Apart from this novel *abstraction* task, the dataset also poses another challenge for NLP systems by requiring them to process long documents. The average number of tokens in our summaries are which is beyond the comfort level of most existing systems. Understanding long narratives and modeling long contexts are new frontiers for NLP research (Roy et al., 2021a; Fan et al., 2021) and `LiSCU` pushes us in this direction. To further facilitate research in this direction, we also release a small dataset where the goal is to read the entire literary piece and generate character descriptions.

We explore the ability of the modern neural models on both tasks. We demonstrate through experiments that although existing models can identify characters reasonably well in masked descriptions, there is still a scope for improvement considering human accuracy on this task. Also, while existing models can generate fluent and logically self-consistent text, they are not always faithful to the literature summaries and fail to capture salient details about the characters. Lastly, we conduct qualitative analysis to provide insights to future works.

4.2 The LiSCU Dataset

We now describe our **Li**terature **S**ummary and **C**haracter **U**nderstanding (LiSCU) dataset. LiSCU is a dataset of literature summaries paired with descriptions of characters that appear in the summaries. Figure 4.1 shows an example of our dataset.

Next, we describe the data collection pipeline for LiSCU (§4.2.1), followed by details on the reproducibility of the data collection process (§4.2.2).

4.2.1 Data Collection and Filtering

We collected LiSCU from various online study guides such as shmoop,[1] LitCharts,[2] SparkNotes,[3] and CliffsNotes,[4]. These sources contain educational material to help students study for their literature classes. These study guides include summaries of various literary pieces as well as descriptions of characters that appear in them. These literature summaries and character descriptions were written by literary experts, typically teachers, and are of high pedagogical quality.

We used Scrapy,[5] a free and open-source web-crawling framework to crawl these study guides. Our initial crawl resulted in a set of $1,774$ literature summaries and $25,525$ character descriptions. These included all characters mentioned in the literary pieces. However, not all characters, especially those that played a minor role in the literary piece, appeared in the corresponding literature summaries. Since our task involves making inferences about characters from the literature summaries, we filtered out the characters which do not appear in the summaries or their names or the descriptions had very little overlap with the literature summaries. This is done to mitigate the reference divergence issue (Kryscinski et al., 2019; Maynez et al., 2020) and ensure that the literature summary has enough information about the character to generate the description. For this, we define the "information overlap" between two pieces of text $\mathcal{A}$ and $\mathcal{B}$, $IO(\mathcal{B}||\mathcal{A})$, as the

[1] https://www.shmoop.com/study-guides/literature
[2] https://www.litcharts.com
[3] https://www.sparknotes.com/lit/
[4] https://www.cliffsnotes.com/literature
[5] https://scrapy.org/

# unique books	1,220
# literature summaries	1,708
# characters	9,499
# characters with accompanying full book	2,052
# unique books with full-text	204
avg. # characters per summary	5.56
min. # characters per summary	1
max. # characters per summary	38
avg. summary length (in tokens)	1,022.32
avg. # sentences in summary	48.82
avg. character description length (in tokens)	184.57
avg. # sentences in description	8.56
# characters in Train set	7,600
# characters in Test set	957
# characters in Validation set	942

Table 4.1: Statistics of the `LiSCU` dataset.

ratio of the length of the longest overlapping word sub-sequence between $\mathcal{A}$ and $\mathcal{B}$, over the length of $\mathcal{A}$.[6] Note that this information overlap measure is not symmetric and intuitively measures how much information about $\mathcal{A}$ is present in $\mathcal{B}$. We used the information overlap measure to filter our dataset as follows. If the information overlap of the literature summary with the character name, $IO(\text{literature summary} \,||\, \text{character name})$, is less than 0.6, then we consider that the character is not prominently mentioned in the literature summary and we remove that character from our dataset. Similarly, if the information overlap between the character description and the literature summary, $IO(\text{literature summary} \,||\, \text{character description})$, is less than 0.2, then we consider the character description generation less feasible and we remove that data point from our dataset.[7]

However, during these filtering steps, we did not want to remove the most important characters of the narrative. The online study guides list characters in decreasing order of their importance in the literary piece. For example, narrators, protagonists, antagonists,

[6] Technically this is the same as Rouge-L precision

[7] These thresholds were chosen by experimenting with different values and manually analyzing the quality of (a subset of) the data.

etc., are always described first. Leveraging this ordering, we always retained the top 3 characters of the literary piece in our dataset.

After the filtering process, our final dataset consists of $1,708$ literature summaries and $9,499$ character descriptions in total. This set was split into train (80%), test (10%), and validation (10%) sets. The data splits were created to avoid any data-leakages – each literary piece and all of its character descriptions were consistently part of only one of the train, test and validation sets. Table 4.1 shows the statistics of the final dataset. The dataset also contains the full-text of the books for $2,052$ of the character descriptions.

4.2.2 Dataset Reproducibility

`LiSCU` is drawn from various study guides on the web. While we do not have the rights to directly redistribute this dataset, to allow other researchers to replicate the `LiSCU` dataset and compare to our work, we provide a simple script[8] to recreate `LiSCU` from a particular time-stamped version of these study guides on *Wayback Machine*, a time-stamped digital archive of the web. Our script ensures that others will be able to recreate the same train, test and validation splits.

4.3 `LiSCU` Task Definitions

We introduce two new tasks on the `LiSCU` dataset:(i) *Character Identification* (ii) *Character Description Generation*.

4.3.1 Character Identification

The *Character Identification* task requires models to identify the character in an anonymized character description. Given a summary S, a candidate list of characters that appear in the literature summary $C = \{c_1, c_2, ..., c_k\}$, and an anonymized character description D^{c*}_{masked}, the goal in this task is to identify the name of the character c^* described in the

[8] https://github.com/huangmeng123/lit_char_data_wayback

anonymized character description. We anonymize character descriptions by masking out all mentions of the character c^* in the original description D^{c*}.

4.3.2 Character Description Generation

The *character description generation* task tests the ability of NLP models to critically analyze the narrative from the perspective of characters and generate coherent and insightful character descriptions. Formally, given a literature summary, S, and a character name, c, the goal in this task is to generate the character's description, D^c. Generating the character description necessitates understanding and analyzing every salient information about the character in the literature summary.

4.3.3 Human Assessment of `LiSCU`

In order to verify the tractability of these two tasks as well as assessing the quality of the collected `LiSCU` dataset, we conducted a set of human evaluations on Amazon Mechanical Turk. We run our human assessment on the full test set of `LiSCU`.

Assessing the Character Identification task: In the first human assessment, we showed annotators the literature summaries, anonymized character descriptions , and a list of character names (plus one randomly sampled character from the literary piece). The descriptions were anonymized by replacing all mentions of the corresponding character names with blanks.[9] For each anonymized character description, we asked 3 judges to identify which character it is describing by choosing from the list of choices. The judges also had the option of saying that they are unable to identify the character given the literature summary and the anonymized character description.

Assessing the Character Description Generation task: In the second human assessment, the judges are shown the same summary along with the original de-anonymized character descriptions. For each character description, 3 judges were asked to evaluate the quality of the description by answering the following two questions:

[9] We identified mentions of a character in the summary by using a coreference system (Joshi et al., 2019b,a) as well as by matching the first name or the full name of the character.

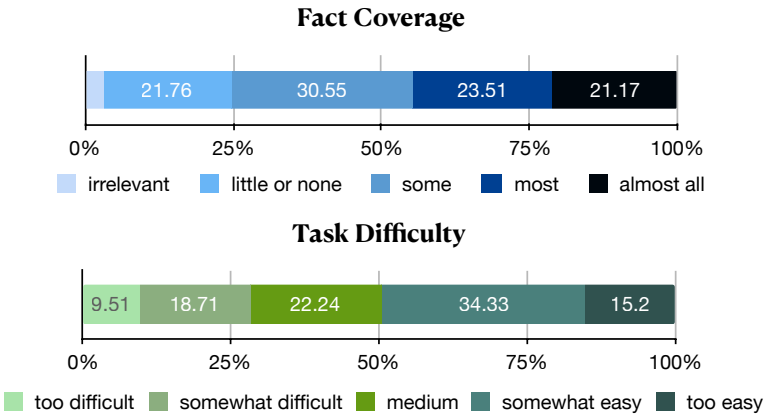


Figure 4.2: Human assessment of the feasibility of the character description generation task.

1. Fact coverage: Specify how much of the information about the specific character in the corresponding "character description" is present in the summary (either explicitly or implicitly). Answer choices included: a) *almost all of the information*, b) *most of the information*, c) *some of the information*, d) *little or none of the information*, and e) *character does not appear in the summary at all*.

2. Task difficulty: Given the summary, how easy is it to write the character description on a Likert scale of 0-4 (0 being too difficult, 4 being too easy)? If in the previous question the judges found that some of the information in the character description was not present in the summary, they are asked to disregard that while answering this question. In other words, they only need to consider the information in the character description which is explicitly or implicitly mentioned in the summary.

We recruited 200 crowd-workers who were located in the US, UK, or CA, and had a 98% approval rate for at least $5,000$ previous annotations. We collected each annotation from 3 workers and use majority vote in our assessments. In the Appendix B.1, we describe several steps we took to alleviate limitations of using crowd-sourcing and ensure high quality annotations. Screenshots of our AMT experiments are provided in the Appendix Figure B.1.

For the first assessment on identifying characters, the human accuracy was 91.80% (Fleiss' Kappa (Landis and Koch, 1977) $\kappa = 0.79$), indicating the feasibility of the task.

For the second assessment of fact coverage and task difficulty, we summarize the result in Figure 4.2. The top chart ('Fact Coverage') shows that around 75% of the of the literature summaries contain reasonable amount of information about the character represented in the corresponding character description. The bottom chart ('Task Difficulty') shows that more than 90% of the times, the human judges considered the task of writing the character descriptions from the literature summaries not too difficult.[10]

These results verify the feasibility of understanding and drawing reasonable inferences about characters in the literature summaries from the `LiSCU` dataset. Next, we describe models and establish baseline performances on the two proposed tasks.

4.4 Character Identification

We present two approaches to address this task: (1) solving it as a multiple-choice classification problem, and (2) using a generative classifier that generates, instead of identifying, the character name, as shown in Figure 4.3.

In the multiple-choice approach, we use the standard setup introduced in BERT (Devlin et al., 2019) where the text from c_i, D^{c*}_{masked} and S (with custom prefix tokens) are concatenated as input, and the `[CLS]` token is projected to a final logit. We apply a Softmax function to the logits to obtain the scores for each c_i. For training practicalities, we limit the number of choices to 4 during training (using the earliest window of choices which include the correct one). During inference, we can generate the logits for all the answer choices since they are independent before the final Softmax.

To establish a baseline performance, we experiment with finetuning RoBERTa (Liu et al., 2019a), and ALBERT (Lan et al., 2020) which have been shown to perform well in several classification tasks. However, both these models cannot process inputs longer than 512 tokens and the concatenated inputs are generally much longer. So we also

[10]There is a natural label bias in the annotations: most of the responses fell into few categories. In this case, standard inter-annotator agreement statistics are not reliable (the well-known paradoxes of kappa (Feinstein and Cicchetti, 1990)). Thus, we simply report a pairwise agreement (i.e., how often do two judges agree on the answer for the same question) of 0.71 and 0.64 for 'fact coverage' and 'task difficulty', respectively.

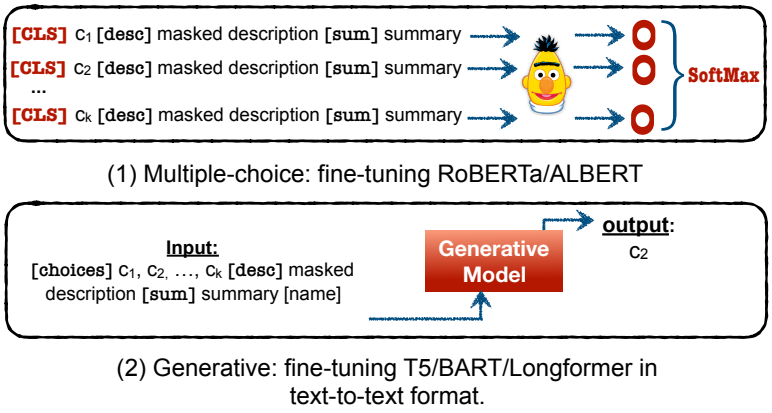


Figure 4.3: Approaches for *Character Identification*.

tried Longformer (Beltagy et al., 2020), a BERT-like model with an attention mechanism designed to scale linearly with sequence length, thus allowing the model to encode longer documents. However, despite trying various hyperparameters, Longformer was not able to match the scores in our experiments.

Our second approach, a generative classifier, is inspired by Raffel et al. (2020a) who studied transfer learning by converting NLP problems into a text-to-text format. The generative classifier addresses the character identification problem by directly generating the character name $\hat{c}$, given all character names (answer choices), the masked character description, and the summary (see Figure 4.3). During inference, we compute the model's probability of each of the answer choices, and output the one with the highest probability.

We use this procedure to train several strong baselines built on top of the following pre-trained transformer-based models: BART (Lewis et al., 2019), T5 (Raffel et al., 2020a), and Longformer (Beltagy et al., 2020).

Implementation Details. The RoBERTa and ALBERT multiple-choice classifiers were trained for 6 epochs, initial learning rate 1e-5 (ADAM optimizer), batch size 16. The generative classifier using BART was trained for 5 epochs, initial learning rate 5e-6, batch size 8. We used the Transformer package (Wolf et al., 2019a) for training. The T5 model was trained for 12 epochs on a TPU using the default parameters from the T5 repository

Model	Description Setup	Accuracy (%)
Random Guess	-	18.70
RoBERTa-Large (Liu et al., 2019a)	partial	77.84
ALBERT-XXL (Lan et al., 2020)	partial	**83.33**
T5-11B (Raffel et al., 2020b)	partial	80.16
BART-Large (Lewis et al., 2019)	partial	74.89
Longformer (Beltagy et al., 2020)	partial	71.10
BART-Large (Lewis et al., 2019)	full	78.58
Longformer (Beltagy et al., 2020)	full	74.78
Human Performance	-	91.80

Table 4.2: Accuracy for the *Character Identification*. The 'partial' description setup used a truncated description (50 words) to allow including more of the summary.

(learning rate 1e-3 with AdaFactor, batch size 8).[11] We truncate the summaries (and descriptions) to satisfy model-specific maximum input length.

Results. Table 4.2 shows the accuracies of different baselines. The highest accuracy is achieved by ALBERT-XXL (83.33%) followed by T5-11B (80.16%). Although both ALBERT and T5 were given partial character descriptions, their specific pre-training loss and larger number of parameters (for T5-11B) lead to superior performance over other baselines. We observe that there is still a significant difference between the human performance (91.80%) and the best model performance (83.33%) on the character identification task, warranting future work on this direction.

4.5 Character Description Generation

We present several strong baselines for generating character descriptions by fine-tuning pre-trained transformer-based language models (LM) (Vaswani et al., 2017). We study two types of models: (1) a standard left-to-right LM, namely GPT2-L (Radford et al., 2019) which is trained with LM objective to predict the next word; and (2) two encoder-decoder models, namely BART[12] (Lewis et al., 2019) and Longformer (Beltagy et al.,

[11] https://github.com/google-research/text-to-text-transfer-transformer

[12] We use the bart-large-xsum as initial weights as our task can benefit from the summarization capability.

2020)[13] which initialize the state of the Transformer by reading the input, and learn to generate the output.

One of the challenges of the proposed task is the length of the summaries, which might exceed the maximum allowable length for most existing pre-trained models. To overcome this, we either: (1) simply truncate the literature summary at the end, or (2) only keep sentences from the literature summary that have a mention of the character of interest. For the latter, we use a coreference resolution model, SpanBERT (Joshi et al., 2019b,a), to identify character mentions within a summary. This results in a modified dataset of character-specific literature summaries paired with character descriptions. In addition to these two approaches, we also fine-tune Longformer (Beltagy et al., 2020) with original full-length literature summary. Longformer leverages an efficient encoding mechanism to avoid the quadratic memory growth and has been previously explored for NLU tasks (encoder-only). We integrate this approach into the pre-trained encoder-decoder BART model to encode inputs longer than its maximum token limit. All the models take `[name]` c `[sum]` S `[desc]` as input and generate the character description D^c as output.

Experiment with Full Literary Pieces. We also run an experiment on a subset of our data with accompanying full-text of the literary pieces. Since it is infeasible to use the full texts as input given the memory constraints of current models, we coarsely select spans of the full-text beginning 50 tokens before, and 50 tokens after the occurrence of character's name. We use a Longformer model where the input is simply the concatenation of the selected spans. Due to the small size of the this subset, we perform a 5-fold cross validation starting from a pre-trained model fine-tuned on summary-description pairs.[14]

Implementation Details. We use the Transformer library (Wolf et al., 2019a). Each baseline was trained for 5 epochs with effective batch size of 8, and initial learning rate of 5e-6. We use the maximum input length of 1024 for GPT2, and 2048 for BART[15] and

[13] `https://github.com/allenai/longformer`. We initialize parameters of Longformer with the same pre-trained BART.

[14] Pre-training data do not contain instances of this subset.

[15] BART originally accepts inputs of maximum 1024 BPE-tokens. We extend this to 2048 by adjusting its positional embeddings.

Model	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BERT-F1
Length Truncated Input					
GPT2-L	0.67	19.25	3.50	17.51	77.71
BART-L	**1.38**	**24.93**	**5.42**	**21.99**	84.54
Longformer	1.05	21.47	4.66	19.37	84.64
Coref Truncated Input					
GPT2-L	0.58	18.69	3.15	16.91	78.46
BART-L	0.96	21.33	4.66	19.04	84.26
Longformer	0.98	21.18	4.40	19.13	84.59
Full Length Input					
Longformer	1.14	21.79	4.88	19.60	**84.72**

Table 4.3: Automatic evaluation results for *Character Description Generation*. BART-L achieved the best BLEU and ROUGE scores while Longformer performed best on BERTScore.

the variant of Longformer with truncated input. For experiment with original books, we use $16,384$ which is the maximum allowable input length for Longformer. During inference, we use beam search decoding with 5 beams.

4.5.1 Automatic Evaluation

Following previous works, we use several standard, widely used automatic evaluation metrics. We use **BLEU-4** (Papineni et al., 2002) that measures overlap of n-gram up to $n = 4$, **ROUGE-*n*** (n=1, 2), and **ROUGE-L** F-1 scores (Lin, 2004).[16] However, recent works (Novikova et al., 2017; Wang et al., 2018) have raised concerns on the usage of these metrics as they fail to capture paraphrases and conceptual information. To overcome these issues, we additionally include a model-based metric, **BERTScore** (Zhang et al., 2020b), which measures the cosine similarity between contextualized embeddings of the gold and generated outputs.[17]

The result of the automatic evaluation is presented in Table 4.3. According to the table, BART-L consistently achieves the best performance across BLEU and ROUGE scores.

[16]Note that we did not include perplexity score as it is not comparable across LM-based and encoder-decoder models.

[17]We use the code at `https://github.com/Tiiiger/bert_score`

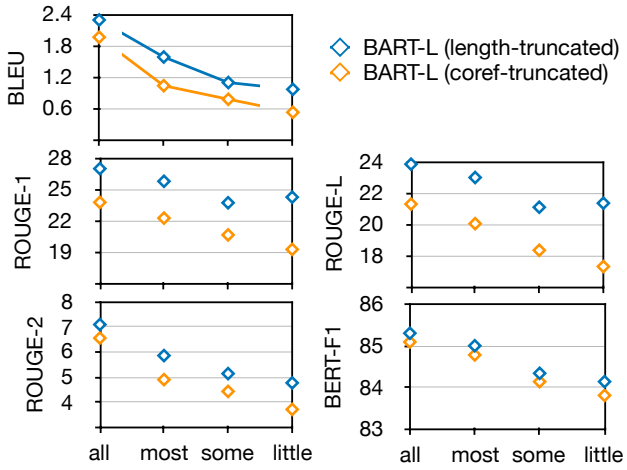


Figure 4.4: Breakdown results for BART-L on subsets with annotated fact coverage as all/most/some/little. Results for other baselines are provided in Appendix.

However, Longformer achieves a slightly better BERTScore. Both BART and Longformer outperform GPT2 in general. This can be in part because BART and Longformer can handle longer context, and are initially pre-trained on a combination of books and Wikipedia data and further fine-tuned on summarization tasks, while GPT2 is pre-trained on WebText only.[18]

Models perform relatively better in the length truncation setups than in the coreference truncation. We posit that this is because a lot of the key points about major characters are likely to appear earlier in the book summary (favoring length truncation). Also, there might be errors introduced by the coreference resolution model itself.

In order to have a better insight into the models' performance with respect to varying level of task feasibility, in Figure 4.4, we additionally report the breakdown of the results for BART-L on separate subsets with "almost all", "most", "some", "little or none" of the information about the character (refer to *Fact Coverage* in §4.3.3). As expected, we observe a consistent decline in the performance with lower amount of fact coverage. Results for other baselines are reported in Table B.1 of the Appendix.

In Table 4.4, we compare the models when using selected spans from the original

[18]While these models could have had access to the original book text, they do not have access to the character descriptions (our outputs) during pre-training. So, this information should not principally change any of our empirical conclusions.

Model	BLEU	R-1/ R-2 /R-L	BERT-F1
Longformer (w/ Books)	0.73	17.61 /3.60 / 16.15	84.33
Longformer (w/ Summaries)	**1.00**	**19.46 / 4.33 / 17.74**	**84.77**

Table 4.4: Automatic evaluation results for models using full-text of books vs. literature summaries.

literary piece as the input vs. literature summaries as the input. We observe a decline in performance when we used the full text. This reveals that even though the literary pieces contain all the character information, this information is scattered which makes it harder for the model to identify important facts about the character. Using full texts also requires encoders which are better at understanding dialog, first-person narratives and different writing styles of the authors. We invite the community to consider this challenging but important problem.

4.5.2 Human Evaluation

To better evaluate the quality of the generated character descriptions, we conduct a human evaluation on 100 test pairs of literature summaries and character descriptions generated by the BART-L model on Amazon Mechanical Turk.[19] Given a literature summary and multiple generated character descriptions (shown one by one), the workers were asked to rate each generated description on a Likert scale of $1-5$ (1 being the worst, and 5 being the best) according to the following criteria: (1) **Grammatical correctness** to indicate if the generated description is grammatically correct, (2) **Logical correctness** to indicate whether the generated description is logically meaningful and coherent, (3) **Faithfulness** of the generated description with respect to the given summary (a faithful character description will not mention facts which are irrelevant to the character and/or not stated in the summary), (4) **Centrality** to evaluate whether the description captures important details and key facts about the character, and finally (5) the **Overall score** considering all the four criteria listed above. We provide a screenshot of the experiment in Figure B.2 of the Appendix.

[19]Here we are evaluating 4 character descriptions per summary, for the total of 25 literature summaries.

Aspects	(1)	(2)	(3)	(4)	(5)
Grammar	0.00	3.67	8.67	28.67	59.00
Logical Corr.	1.67	9.00	19.00	33.67	36.67
Faithfulness	12.67	23.67	21.00	24.67	18.00
Centrality	15.33	17.00	27.00	23.67	17.00
Overall	11.00	19.33	26.67	26.67	16.33

Table 4.5: Percentage of different ratings from human evaluation of generated descriptions (1=worst, 5=best).

Figure 4.5 presents the results of this human evaluation. We observe that the generated descriptions show a reasonable level of grammatical (4.43) and logical correctness (3.94). However, they lack behind when it comes to faithfulness (3.11) and centrality (3.10). We also report the distribution of ratings in Table 4.5. These results indicate that solving this task requires designing better models of character-centric analysis of narrative.

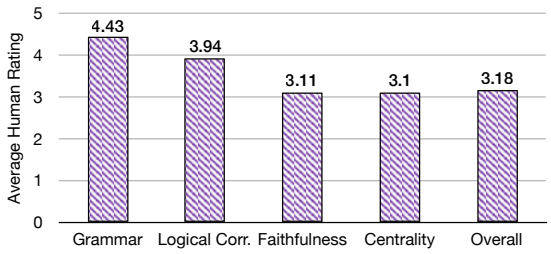


Figure 4.5: Human evaluation of generated character descriptions. While the descriptions are grammatically correct and logically coherent, they often misrepresent or miss important details about the character.

Here, we do a qualitative analysis for the *Character Description Generation* task. In our human evaluation of the generated character descriptions (§4.5.2), we additionally provided a questionnaire to collect in-depth feedback from crowd-workers on the type of errors the BART-L model made when generating character descriptions from the given literature summaries. The questionnaire asked "What details about the character does the given character description miss or describe inaccurately. Note that the description is supposed to describe only the important details and not necessarily all of them." The workers were asked to select all the applicable choices among the following error types:

1. **Events:** The character description misses or misrepresents some main event(s) that the character is involved in.

2. **Role:** The character's role in the narrative (e.g., protagonist, antagonist, etc.) is

important but is not included or misrepresented in the character description.

3. **Relationships:** The character's relationship with other characters is important (e.g., the protagonist's wife) but is not included or misrepresented in the character description.

4. **Personal characteristics:** The character's personal characteristics (e.g., age, ethnicity, personality, etc.) are important for the narrative but are not included or misrepresented in the character description.

5. **Behavioral characteristics:** The character's motivation, desires, and behavior are important but are not included or misrepresented in the character description.

6. **No major error:** None of the above. The character description captures most of the important details about the character.

We also provided an optional text box for them to type in other details that are missing or misrepresented but not listed above.

The result of this analysis is shown in Table 4.6. We can see that the generated descriptions make fewer mistakes in capturing personality-related attributes (12.33%) and more mistakes in representing important events involving the characters (46%). They also sometimes omit or misrepresent roles (24%), relationships (25%), and behavioral characteristics (22%) of the characters. This indicates factors that future systems should consider improving upon when addressing this task.

Error Type	Percentage
Events	46.00
Role	24.33
Relationships	25.00
Personal characteristics	12.33
Behavioral characteristics	22.33
No major error	27.67

Table 4.6: Error Analysis: proportion of generated descriptions with different error types.

We provide qualitative examples of the generated character descriptions along with the errors they made (as pointed out by the turkers) in Table 4.7. More examples with input literature summaries are provided in Tables B.2 to B.5 of the Appendix.

Generated Character Descriptions	Comments
Book title: The Three Sisters **Character name:** Vershinin **Generated Character Description:** Vershinin is one of the most interesting characters in Three Sisters. He's a young officer in the Russian army, and he's got a lot of time on his hands. But that doesn't stop him from having an affair with Irina, the youngest sister, who falls in love with him because she's attracted to his philosophical ideas about the meaning of life.	According to the summary "Masha", the middle sister, is the one who has an affair with the officer Vershinin, and not the youngest sister, Irina.
Book title: The Sun Also Rises **Character name:** Pedro Romero **Generated Character Description:** Pedro Romero is a young Spanish bull-fighter who is in love with Lady Brett Ashley, a beautiful British divorcee. He's a talented and talented young man, and, despite the fact that he is only twenty-five years old, he has the potential to be one of the best bullfighters in Pamplona. Jake feels terrible for introducing him to Brett, fearing that it has corrupted him.	The description captures most of the important details about the character.

Table 4.7: Examples of generated descriptions. Words in red correspond to hallucinated or missing content, and words in green correspond to faithful information. The input literature summaries are provided in the Appendix.

4.6 Related Works

The field of computational narrative understanding studies how to algorithmically represent, understand, and generate stories. Early computational studies on narratives had focused on learning procedural scripts and event sequences (Schank and Abelson, 1977; Manshadi et al., 2008; Regneri et al., 2010), narrative chains or schemas (Chambers and Jurafsky, 2008, 2009), and plot units (Goyal et al., 2010; McIntyre and Lapata, 2010; Elsner, 2012).

Computational linguists have also worked on character-centric modelling of narratives (Chambers, 2013). The character-centric perspective aims to understand characters – their personas, roles, goals, relationships, emotions, etc. Previous works have proposed methods to detect characters and infer latent personas in movie plot summaries and fictional novels (Bamman et al., 2013b, 2014b; Vala et al., 2015; Flekova and Gurevych, 2015), model inter-character relationships (Iyyer et al., 2016; Srivastava et al., 2016; Chaturvedi et al., 2017a; Kim and Klinger, 2019b), and emotions (Brahman and Chaturvedi, 2020).

Earlier works have also considered constructing social networks of characters (Agarwal et al., 2014) from novels (Elson et al., 2010; Elsner, 2012) and films (Krishnan and Eisenstein, 2015b).

Another line of work related to ours is on summarization of novels (Mihalcea and Ceylan, 2007). This work built a dataset of novel-summary pairs and used unsupervised summarization models such as *TextRank* (Mihalcea and Tarau, 2004) and *MEAD* (Radev, 2001). Instead of summarizing full novels, Ladhak et al. (2020) proposed a content-selection approach to create a gold-standard set of extractive summaries by aligning chapter sentences with abstractive summary sentences.

In a more related work, Zhang et al. (2019) collected a dataset of fictional stories along with author-written summaries. They proposed an extractive ranking and a classification approach to select a subset of salient attributes from a list of candidate attributes (extracted from the story) that describe a character's personality. While this work presented a collection of personality-related phrases as a potential summary for the actual novel, our dataset contains literature summaries and character descriptions, and we aim to generate natural language texts that analyze the narrative from the perspective of the characters. Such an analysis is more in-depth than a collection of phrases.

4.7 Summary

Character(s) are principle narrative elements, and comprehending them has been widely recognized as key to understanding narratives. Human readers build a mental model of characters, understand what they look like, their role in the literary piece, and assess their psychology, motivations, and consequences of their behavior. However, building such a deep understanding of fictional characters in narratives is hard for machine reading systems. To encourage progress in character-centric understanding of narratives, we present `LiSCU`, a dataset of literature summaries paired with descriptions of characters that appear in them. We use `LiSCU` to propose two tasks that explore the ability of the modern neural models to understand the narrative from the perspective of characters.

Performing human assessments on the model outputs show that there is still much room for improvement on these tasks.

Chapter 5

Entity Descriptor with Guiding Keys

In the previous chapter, we studied characters as the elements important for understanding narratives and proposed to generate character descriptions grounded in fictional stories. In this chapter, motivated by the Centering theory (Grosz et al., 1995), we study entities (and not just characters) as the narrative element. Specifically, we propose a real-world application of generating narratives about entities grounded in world knowledge. Towards this, we present a new dataset EntDeGen, ranking-based generation models for this task, and an evaluation metric for factual correctness MAFE.

5.1 Introduction

Entities and entity knowledge have been shown to play a crucial role in many NLP applications including open-domain question answering (Xu et al., 2016; Das et al., 2017), summarization (Amplayo et al., 2018), contextualized knowledge representations (Peters et al., 2019), and dialogue and narrative generation (Wu et al., 2021; Clark et al., 2018b).

In this chapter, we propose a new entity description generation task which is an instance of grounded keys-to-text generation. Given an entity name, a wishlist of *keys* (without the actual values) along with a set of short *grounding passages*, the task is to generate a factual description about that entity. The wishlist of *keys* acts as guiding keys to control the knowledge and facts the model should generate. An example is shown in

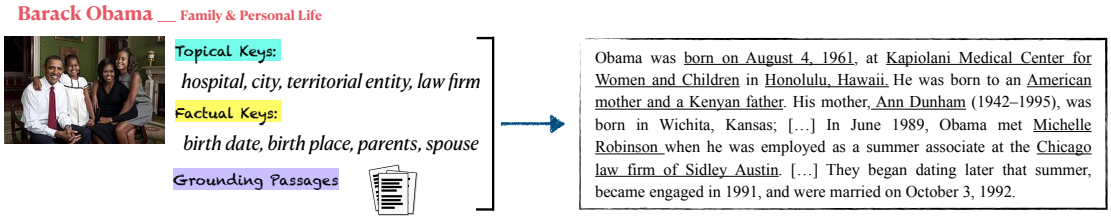


Figure 5.1: An example from EntDeGen dataset and entity description generation task. Given a set of topical and factual keys, along with multiple grounding passages, the task is to generate an entity description. Corresponding knowledge are underlined.

Figure 5.1, where the task is to generate a paragraph about "Barack Obama", in particular about his family and personal life. Potential *factual keys* in this example are "birth date, birthplace, parents, spouse, children", etc. The task also enables a finer-grained control over the types of entities to be included in the output via *topical keys* such as "hospital, city, law firm" for the example in Figure 5.1. Finally, pertinent information about the entities needs to be fetched from the set of *grounding passages* which are short snippets of text.

Entity description generation can be utilized in voice search where the retrieved knowledge needs to be converted into text and read to the user. This task also has applications in other AI systems when used to verbalize knowledge for pre-trained language model (PLM) training (Ma et al., 2021), or incorporating knowledge entity as additional input to models for Commonsense QA (Xu et al., 2021).

Our proposed task differs from similar existing tasks, such as data-to-text generation (Cheng et al., 2020; Puduppully et al., 2019) in that, we presume keys but not values are given. This is a more realistic setting as knowledge about entities are time-dependent and changing constantly, and so have to be fetched on the fly. Our proposed task also differs from the prompt-to-text generation, where only the entity is given as prompt. Prompt-based methods are open-ended, and their lack of controllability leads to an output that does not meet the user's needs. They are also prone to hallucination due to not being grounded enough in factual knowledge.

To facilitate research on grounded keys-to-text generation task, we introduce a large-scale and challenging dataset, called EntDeGen, with about 375K instances. The *grounded,*

factual and *long-form* nature of the task brings a new challenge, i.e., generating paragraph-level text which is faithful to some sources of knowledge based on the provided guiding keys. To address this challenge, we propose EntDescriptor equipped with strong ranker to help the model focus on passages that are both relevant to the keys and complementary. We propose two rankers. Our contrastive dense ranker is based on embedding-based retrieval systems trained in a contrastive framework. Our autoregressive ranker generates a sequence of passage indices autoregressivly by modeling the probability of each passage conditioned on previously generated passages. This ranker is shown to achieve the strongest performance by modeling the joint probability of passages.

The factual aspect of generation also calls for a new evaluation metric. Inspired by recent fact-based evaluation for summarization, we propose an automatic metric, called MAFE, to evaluate different aspects of grounded text quality, including relevance and consistency.

5.2 Task: Entity Description Generation

Given an entity e, title t, a set of factual $\mathcal{K}^f = \{k_1^f, k_2^f, ..., k_m^f\}$ and topical $\mathcal{K}^t = \{k_1^t, k_2^t, ..., k_m^t\}$ keys, and grounding passages $\mathcal{P} = \{p_1, p_2, ..., p_N\}$, the goal is to generate a text h with respect to the provided keys (see Figure 5.1).

5.3 Dataset: EntDeGen

Our dataset collection strategy is motivated by the WikiTableT dataset (Chen et al., 2021) and is based on Wikipedia. Each Wikipedia article $\mathcal{A}_w$ is composed of multiple sections $\mathcal{S} = \{s_1, s_2, ..., s_n\}$. For example, a typical Wikipedia article about a football player may contain section about "Introduction", "Early Life", "Club Career", and "International Career". The title of the Wikipedia article is the entity whose description is to be generated and each section form the reference description r. We perform the following steps for each section s_i in a Wikipedia article to obtain: factual keys, topical keys, and grounding passages.

Factual Keys. For obtaining factual keys, we align key-value pairs in infobox and Wikidata with each section s_i. For this, we We took a distant-supervision approach to estimate the alignment score of each key-value pair with the section using semantic similarity and lexical precision. For semantic similarity, we compute the precision component of BERT-Score (Zhang et al., 2020b) between the section text and the concatenation of key-value pair (`key + value`). A high value indicates that the key-value pair is semantically relevant to that section. We also measure the ROUGE-L precision score (Lin, 2004) between the section text with respect to the concatenation of key-value pair. For each instance in our dataset, we select keys whose key-value BERT-Score is greater than 0.82, and ROUGE-L score is greater than 0.25.[1]

Topical Keys. For obtaining topical keys, we first find all hyperlinked articles $\mathcal{A}_h$ appearing in the section. We then use the value of the "instance of" or "subclass of" tuple in the Wikidata table of $\mathcal{A}_h$ as the set of topical keys for section s_i. For example, the hyperlinked *Kapiolani Medical Center for Women and Children* in Figure 5.1 is an instance of *hospital* according to its Wikidata table. So it will be turned into a topical key *hospital*.

Grounding Passages. For obtaining grounding passages, we use the documents in the WikiSum dataset (Liu et al., 2018). The documents are cited references in the Wikipedia article obtained by CommonCrawl or web pages returned by Google Search. Each instance in our data has 40 grounding passages. Note that our dataset is distantly supervised, and these passages may not always contain all the facts regarding the keys. To enhance the quality of our dataset, we filter out entities for which the average Bert-Score recall of key-value pairs against the grounding passages is lower than 0.82.[2]

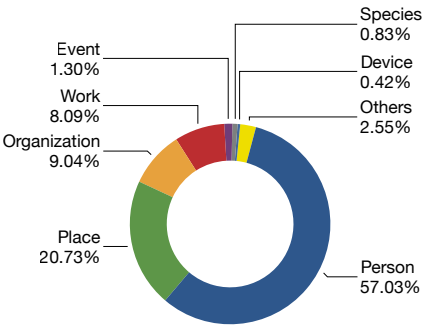


Figure 5.2: ENTDEGEN Entity Domain Distribution.

Basic statistics of ENTDEGEN is provided in Table 5.1. Figure 5.2 depicts the diversity

[1]These threshold values are selected empirically from BERT-Score $\in \{0.80, 0.82, 0.84, 0.86\}$ and ROUGE-L $\in \{0.20, 0.25, 0.30\}$.

[2]Similarly, this value is chosen empirically from BERT-Score $\in \{0.80, 0.82, 0.84, 0.86\}$

	Train	Dev	Test
Instances	267,453	2,500	2,497
Avg. Output Len. (token)	116.37	125.56	124.10
Avg. Passage Len. (token)	59.98	60.13	60.25
Avg. Top. Keys	4.14	4.15	4.30
Avg. Fac. Keys	6.04	6.13	6.17

Table 5.1: EntDeGen Dataset Statistics.

of entity domains in EntDeGen while still a large portion of it is Person (character). We associate each entity in our dataset with a domain such as Person, Place, Organization, Event, etc. using DBPedia knowledge-base (Lehmann et al., 2015).

5.4 MAFE: Multi-Aspect Factuality Evaluation

To safely deploy models trained on this task in real-world scenarios, it is crucial to evaluate the factuality of the generated texts. Inspired by recent fact-based evaluation in abstractive summarization (Scialom et al., 2019; Durmus et al., 2020; Wang et al., 2020), we propose to assess the factuality of generation through question answering (QA). We evaluate factuality of a generated description h with respect to (i) factual triples (e, k, v) which are constructed from the entity e, each factual key k and its value v[3], and (ii) reference (gold) description r. In our QA based Multi-Aspect Factualy Evaluation (MAFE), questions are generated from spans in the reference and factual triples (recall), or the generated output (precision), and are automatically answered using the output, or reference-factual triples. Then, the similarity between the predicted answer and the gold answer is used to compute recall and precision. Our evaluation framework is illustrated in Figure 5.3 which accounts for both relevance (recall) and consistency (precision):

Recall ($h \rightarrow r$) evaluates the generated output h on recalling information from the factual triples (e, k, v) AND reference r. For this, we generate questions that have gold answers in factual triples and reference using a Question Generation (QG) module, and

[3]This resembles the *(s, r, o)* in the Knowledge Bases, e.g., `(Barack Obama, place of birth, Hawaii)`.

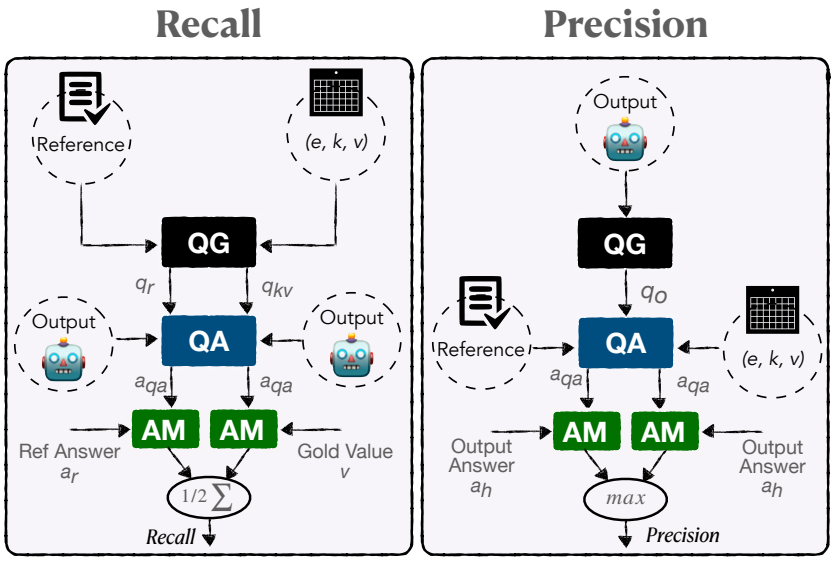


Figure 5.3: MAFE Evaluation Framework consisting of Question Generation (QG), Question Answering (QA) and Answer Matching (AM) components.

obtain answers to these questions from generated output h using a Question Answering (QA) module. We define recall as the average scores of these answers when compared to the gold answers (computed by an Answer Matching (AM) module).

Precision $(r \rightarrow h)$ measures the amount of information in the generated output h that is consistent with factual triples (e, k, v) OR reference r. For this, we generate questions from output and obtain answers from factual triples and reference. We define precision as the maximum score between answers predicted from factual triples and reference.[4]

Next, we describe the 3 modules of the evaluation framework.

5.4.1 Question Generation

Given a sentence s and an answer a, we train a QG model to generate a question q, modeling $P_{qg}(q|s, a)$. For evaluating a generated output, we gather a set of answer spans a by extracting all name entities and noun phrases from each sentence s (of reference or output) using spaCy. For generating questions from factual triples, we linearize them by concatenating their constituent elements and consider the value v as the answer

[4]We use *maximum* because a fact contained in the model's output should either be precise w.r.t knowledge triples or the reference, but not necessarily both.

a. For example, we form "*Barack Obama place of birth hawaii*" from `(Barack Obama, place of birth, Hawaii)`. Following (Durmus et al., 2020), our QG model is a BART model fine-tuned on (s, a, q) triples annotated by Demszky et al. (2018).

5.4.2 Question Answering

Given a question q, and a context c, the QA model gives the probability of an answer a, modeling $P_{qa}(a|q, c)$. For evaluating a generated output, given a question q generated by the QG model from the reference and factual triples, or the output, the QA model answers it using the output, or reference and factual triples (as context c), respectively. For answering questions using factual triples as context, we concatenate all the linearized triples into a single piece of text. Our QA model is an ALBERT-XL model (Lan et al., 2020) fine-tuned on SQUAD2.0 (Rajpurkar et al., 2018). SQUAD2.0 support identifying *unanswerable* questions which is crucial since not all answers are contained within a given context c.

5.4.3 Answer Matching

The common approach to assess the answers given by a QA model (compared to gold answers) is to use F1-score, which is based on exact matching of n-grams. We argue that is problematic in our case when correct answers are lexically different. For example, Sport:"*professional wrestling*" can be realized as "*She is a wrestler* [...]". The F1-score does not capture these lexically varied but correct answers. Therefore, we propose using an NLI model to compare the similarity of two answers. Given the generated question q from the reference, to compare the reference (gold) answer and the predicted answer, we concatenate each answer with the question separately to form the premise and hypothesis for the NLI model. For example, for the question *'What sport did Mr. Kenny Jay play?"*, we pass the following to the NLI model:

`Premise:` *What sport did Mr. Kenny Jay play? professional wrestling*

`Hypothesis:` *What sport did Mr. Kenny Jay play? wrestler*

We give the predicted answer a score of 1 if the NLI model predicts entailment, and a

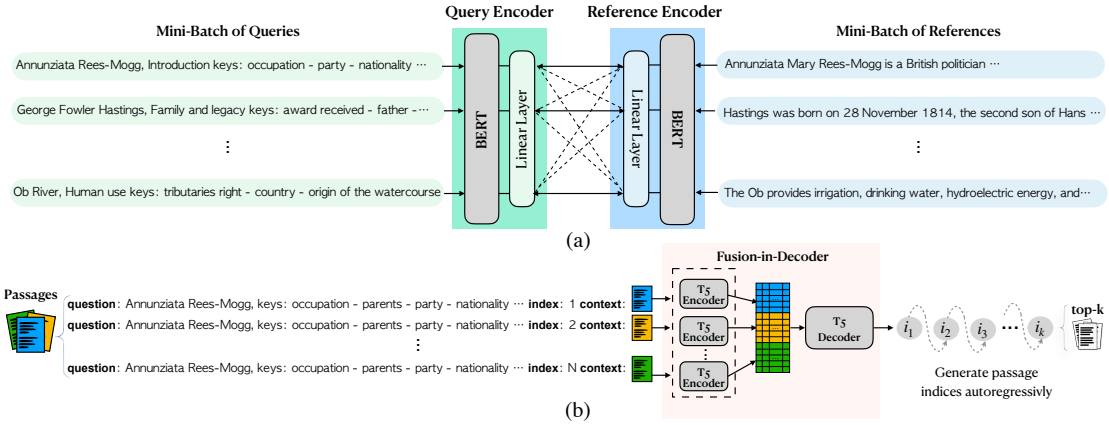


Figure 5.4: Two proposed passage rankers: (a) Contrastive Dense (b) Autoregressive.

score of 0 if it predicts contradiction. For neutral, we compute the BERTScore (Zhang et al., 2020b) comparing the contextualized representations of the two answers. For the NLI model, we use RoBERTa (Liu et al., 2019b) fine-tuned on MNLI (Williams et al., 2018).

5.5 EntDescriptor

The EntDescriptor model needs to fetch relevant passages (source of knowledge) on the fly to generate a factual description. For this, we equip our EntDescriptor model with a *Passage Ranker* (§5.5.1). Given the entity, keys and a set of ranked passages, the *Descriptor Generator* (§5.5.2) then generates an entity description.

5.5.1 Passage Ranker

Each instance in our dataset is accompanied by a set of candidate grounding passages. However, not all passages contain useful knowledge about certain aspects of an entity, i.e., the provided *factual* and *topical keys*. We, therefore, introduce a ranking stage where we rank the grounding passages $\mathcal{P}$ given the entity, title, and a set of keys as query q. The ranker outputs top-k passages $\{p1, ..., p_k\} \subset \mathcal{P}$, which are then used to ground the Descriptor Generator. Below, we describe two baseline rankers, namely ROUGE-2 and tf-idf rankers, and two proposed rankers, namely contrastive dense and autoregressive

rankers.

ROUGE-2 (*oracle*). This ranker ranks passages according to their ROUGE-2 recall against the reference. This is akin to *oracle* ranking as we use information in the reference to do the ranking.

Tf-idf. This ranker ranks passages using their tf-idf score following (Liu et al., 2018).

Contrastive Dense. This ranker learns and then compares dense representations of queries and passages using contrastive training. We train a dense ranker (shown in Figure 5.4(a)) which is inspired by recent embedding-based retrieval systems such as REALM (Guu et al., 2020), and DPR (Karpukhin et al., 2020). We follow a distant supervision approach (Jernite, 2020) by using the reference descriptions r instead of the gold passages as supervision signal. For each instance, we form a query q_i by concatenating the entity, title and the set of keys. We thus construct a dataset of (q_i, r_i) pairs and use a bi-encoder architecture to project queries and references to 128-d embedding space. We use a contrastive framework with in-batch negatives where the idea is to push encoded vector of a query closer to its corresponding reference vector, but away from other reference vectors in the batch. Formally, we optimize the following Cross-Entropy loss with in-batch negatives:

$$\mathcal{L} = -\sum_{(q_i, r_i) \in mB} \log \frac{\exp(\mathbf{q_i}.\mathbf{r_i})}{\sum_{r_j \in mB} \exp(\mathbf{q_i}.\mathbf{r_j})} \tag{5.1}$$

where $\mathbf{q_i}$ and $\mathbf{r_i}$ are encoded query and reference vectors, and mB denotes the mini-Batch. We use mini-batches of 1024, and initialize the encoders with distilled-BERT (Turc et al., 2019; Devlin et al., 2019). Two projection layers are then learned for queries and references. Once the ranker is trained, we use the reference encoder to encode each grounding passage p_i and score them based on their dot product similarity w.r.t vector representation of query $\mathbf{q_i}$. We then use the top-k passages as input to Descriptor Generator.

Autoregressive. In the previous ranker, passages are scored independently according to their relevance to the input query q. However, an ideal ranker should select relevant

yet diverse passages. To achieve this goal, we develop an autoregressive ranker with an encoder-decoder architecture (shown in Figure 5.4(b)) where the encoder process the entire set of passages $\mathcal{P}$, and the decoder generates a sequence of k passage indices. The autoregressive nature enables modeling the joint probability of passages $P(p_1, ..., p_N|q)$. Similar text-to-index framework showed promising results for *sentence ordering* (Basu Roy Chowdhury et al., 2021). To enable encoding the entire set of passages (in our case 40), we use the Fusion-in-Decoder (FiD) architecture following Izacard and Grave (2021).

The FiD architecture takes the input query (concatenation of entity, title and set of keys) as well as each individual passage independently as inputs to its encoders. The query is concatenated with each passage and its positional index using special tokens: `question: [Entity]` e `[Title]` t `[Keys]` $\mathcal{K}^f + \mathcal{K}^t$ `index:` i `context:` p_i. The encoders output hidden representations $\mathbf{h_i} \in \mathbb{R}^{L \times d}$ for each individual passage p_i, where L, and d are input length and hidden dimension, respectively. The concatenation of encoders' representations $\mathbf{H} = [\mathbf{h_1}; ...; \mathbf{h_N}] \in \mathbb{R}^{L.N \times d}$ is passed to decoder which in turn generates a sequence of passage indices $\{i_1, ...i_k\}$. The corresponding top-k passages $\{p1, ..., p_k\}$ are then used to ground the Descriptor Generator. All encoders and decoder are initialized with pre-trained T5 (Raffel et al., 2020c). This ranker is trained using the *silver* sequence of passage indices obtained by ROUGE-2 (*oracle*) ranker.

5.5.2 Descriptor Generator

Extractive. We build an extractive baseline using QG and QA models. For this, we convert an entity name and each factual key in our input into a natural language question using a seq2seq model. We then use a strong extractive QA model, namely ALBERT-XL fine-tuned on SQUAD2.0, to answer these questions using all grounding passages as the context.[5] Finally, we concatenate all sentences from the groundings which contained the most confident answers as our final output.

Abstractive. We build strong abstractive baselines by fine-tuning several transformer-

[5]Each grounding passage is passed separately.

based PLMs. This include encoder-decoder models, namely **BART-Large** (Lewis et al., 2020a), **T5-large** (Raffel et al., 2020c), and **PEGASUS** (Zhang et al., 2020a). All baselines take inputs in the form of `[Entity]` e `[Title]` t `[Keys]` $\mathcal{K}^f + \mathcal{K}^t$ `[docs]` P^k and generate the entity description.

Note that during training our generation models, we use the top-10 grounding passages obtained by *oracle* ROUGE-2 ranker.[6] Implementation details are provided in Appendix C.1.

5.6 Experiments

5.6.1 Automatic Evaluation

We use several widely used automatic metrics like **BLEU** (Papineni et al., 2002), and **ROUGE-L** (Lin, 2004). We also use **PARENT** (Dhingra et al., 2019)[7] that considers similarity of generation to both data (in our case factual triples) and the reference. Lastly, we use **BERTScore** (Zhang et al., 2020b) which computes alignment between BERT representations of reference and generated output.

5.6.2 Evaluation of MAFE

We propose a metric for evaluating factuality, MAFE. To evaluate MAFE as a metric, we compute its correlation with human judgments of factuality. We take a random set of 297 BART-L generated outputs using different rankers. The instances include diverse set of entity domains (see Figure C.1 in Appendix). We collect human judgments of factuality on this subset using Amazon Mechanical Turk (AMT). Three annotators judged the recall-oriented and precision-oriented factuality of each generated paragraph. For evaluating recall-oriented factuality, we present each sentence of the reference one at a time and ask annotators how well the sentence is supported by the content in the generated paragraph. The annotators have to choose from a Likert scale of 1-5 (1 being

[6] Using the passages obtained from other rankers during training degrades the performance.

[7] We use the co-occurrence version which is recommended when paraphrasing is involved between data and text.

Metric	BLEU	R-L	PAR	BERT-S	MAFE
Corr.	48.00	56.75	40.80	56.76	**60.14**

Table 5.2: Paragraph-level Pearson correlation coefficient between automatic metrics and human judgement of factuality. All correlations are significant at $p < 0.001$.

very badly supported, 5 being very well supported).[8] We also present each factual triple one at a time and ask annotators if it is supported by the content in the generated paragraph. [9] For evaluating precision-oriented factuality, we switch references and factual triples with generated paragraphs, i.e., we show the generated paragraphs one sentence at a time and ask how well the sentence is supported by the reference and all factual triples. We then average scores across all sentences. See Appendix C.2.2 for details and screenshots of annotation layout.

To account for recall and precision oriented values, we measure correlations between human judgment F1 ($2\frac{rec.prec}{rec+prec}$) with MAFE-F1 and other automatic metrics. According to Table 5.2, MAFE shows a higher correlation with the human judgment than other metrics. Hence, we include MAFE in our experiments to gauge the factuality of generations.

5.6.3 Results

Performance of Different Baselines. Table 5.3 reports the performance of different baselines for the task of entity description generation. According to the results, Extractive performs poorly compared to other abstractive baselines. This is mainly because it lacks the narrative flow required for a coherent output. Comparing all abstractive baselines, when they are given *oracle* groundings (defined in §5.5.1), shows that BART outperforms T5 and PEGASUS in general on all n-gram overlap-based, PARENT, as well as BERTScore metrics.[10]

[8] We find the Likert scale to be more suitable than binary decision because each sentence might contain multiple facts.

[9] Factual triples are presented in the form of *e (k; v)*. E.g. *Henry Stanton (placeofburial; West Point)* which is read as "The place of burial of Henry Stanton is West Point."

[10] Uni/bi-gram overlap (R-1,R-2) are reported in Table C.1 of Appendix.

Model +Ranker	**BLEU**	**ROUGE-L**	**PARENT**	**BERT-Score**	**MAFE**	
					Recall	Precision
Extractive	2.94	19.85	14.22	82.44	19.01	18.91
T5-Large	5.41	15.09	20.58	84.25	17.65	17.38
+Tf-idf	5.50	26.70	21.42	84.53	17.68	18.35
+Contrastive Dense	6.89	28.14	22.44	84.92	19.00	19.66
+Autoregressive	7.24	28.64	23.45	84.96	**19.48**	19.97
+*Rouge2 (oracle)*	8.56	30.84	25.97	85.41	20.99	22.01
BART-L	7.03	30.82	23.57	86.10	17.43	23.13
+Tf-idf	6.97	31.14	23.96	86.23	17.45	24.02
+Contrastive Dense	8.25	32.46	24.67	86.48	18.55	26.00
+Autoregressive	**8.69**	**32.97**	**25.40**	**86.58**	19.11	**26.71**
+*Rouge2 (oracle)*	9.82	34.87	27.25	87.01	20.28	29.32
PEGASUS	6.49	27.11	22.88	83.65	15.34	22.72
+Tf-idf	6.34	27.25	22.68	83.74	14.79	23.72
+Contrastive Dense	7.99	28.94	24.10	84.43	16.40	24.70
+Autoregressive	8.55	29.75	25.38	84.54	17.16	25.34
+*Rouge2 (oracle)*	10.05	31.71	27.72	84.97	18.07	26.73

Table 5.3: BLEU, ROUGE-L, PARENT, BERT-Score, and MAFE scores for different unranked models, as well with adding different rankers. Models consistently perform better when using autoregressive ranker.

When comparing baselines with respect to factuality using our MAFE metric, we see that BART in general generates paragraphs that are significantly more consistent (precise) with respect to factual triples and reference. Whereas, T5 is slightly better at content-selection (measured by recall).

Performance of Different Rankers. We now investigate the effect of different rankers on generation performance. For this, we compare baselines using different rankers (see Table 5.3). All models perform better when they are given top-k ranked groundings than their *Unranked* baselines. For all generation models, the proposed contrastive and autoregressive rankers significantly outperform the tf-idf baseline ranker. This is because tf-idf ranker only finds passages that feature sparse words from the input query and fails to capture semantic similarities. Moreover, by predicting a sequence of passages each conditioned on the previously selected passages in the autoregressive

Ranker	Recall@5	Recall@10
Tf-idf	32.02	42.35
Contrastive Dense	36.62	45.90
Autoregressive	**44.67**	**52.08**

Table 5.4: Recall@k (%) for different rankers w.r.t *oracle* ranking.

ranker, the generation model gains further improvements over the strong contrastive dense ranker.We also measure Recall@*k* score for different rankers with respect to the *oracle* ranking using ROUGE-2. The score indicates the proportion of *oracle* passages (obtained y ROUGE-2 method) that is found in the top-*k* predicted passages by any of the rankers. Results are shown in Table 5.4 where we observe that autoregressive outperforms the other two rankers.

5.6.4 Human Evaluation

Factuality $(r \leftrightarrow h)$. We evaluate the factuality of generated paragraphs using human annotators. We randomly sample 100 datapoints from the test set and evaluate paragraphs generated by BART-L using four rankers: tf-idf, contrastive dense, autoregressive and ROUGE-2 (*oracle*) (a total of 400 generation examples). We ask 3 judges from AMT to evaluate the recall-oriented and precision-oriented factual correctness of each sample generation. We use the same annotation layout described for evaluating MAFE metric (correlation analysis; §5.6.1). More details can be found in Appendix C.2.2 .

Table 5.5 shows that human annotators consistently rate the factuality of paragraphs generated using autoregressive ranker higher than those generated using contrastive dense ranker and lower than *Oracle* ranker. The result is consistent with our proposed metric as well.

Faithfulness $(P^k \rightarrow h)$. We also evaluate whether the generated outputs are faithful to the top-k grounding passages.[11] For this, we randomly sample 100 data points from the

[11] Here, we are evaluating faithfulness wrt input groundings. Thus, we use the same set of groundings (by fixing the ranker to be autoregressive) and evaluate different underlying LM.

Ranker	Recall	Precision
Tf-idf	48.95	43.36
Contrastive Dense	51.98	57.50
Autoregressive	**56.76**	**58.41**
Rouge2 (oracle)	58.92	62.00

Table 5.5: Human evaluation of factuality (recall- and precision-oriented in %) for BART-L generated paragraphs using different rankers.

	T5-Large	BART-L	PEGASUS
Human Rating	**4.17**	3.53	3.83

Table 5.6: Human evaluation of faithfulness of different baselines w.r.t grounding passages. Scores are on a scale of 1 (very poor) to 5 (very high).

test set and ask 3 annotators from AMT to evaluate the faithfulness of generated outputs using different baselines on a scale of 1-5. Following our previous annotation layout, we show one sentence at a time and then average scores across all sentences. Table 5.6 shows that T5 generates more faithful paragraphs compared to other baselines.

5.6.5 Ablation Studies

Here, we discuss different ablations where we remove/add certain information from/to the input and investigate its effect on the performance. We experiment with settings where there are *no groundings*, *no keys*, *no factual keys*, *no topical keys*, *values w/o groundings*, and *values w/ groundings*.

Table 5.7 shows the results for the BART-L baseline with the autoregressive ranker. As expected, the model performance degrades the most w.r.t all metrics when the grounding passages are removed from the input. This setting is similar to the prompt-to-text generation, where the model mostly relies on its parametric knowledge and is prone to hallucination. Removing all the keys from the input is detrimental in recalling important information, as shown from the MAFE-R score. We also observe that ablating

Ablated Inputs	R-L	BERT-S	MAFE	
			Recall	Precision
Grounding Passages				
no groundings	25.44	84.80	8.87	13.01
Keys				
no keys	30.34	85.95	17.99	27.12
no factual keys	**31.92**	**86.35**	18.21	27.14
no topical keys	31.67	86.33	**19.13**	**28.15**
Orig. Task Input	32.97	86.58	19.11	26.71
Values & Grounding Passages				
values w/o groundings	*28.82*	*85.90*	*19.30*	*25.97*
values w/ groundings	*33.61*	*86.70*	*21.77*	*29.40*

Table 5.7: Ablation study: Best results are in bold. The gray section is when values are assumed to be at hand, and akin to *oracle* experiment.

factual keys hurts the relevance of the generated paragraph (i.e., Recall) w.r.t its reference more, whereas ablating topical keys hurts the n-gram overlapping metric (R-L). This is because factual keys are essential to make a good content selection and be rewarded by MAFE metric, whereas topical keys mostly appear verbatim in the output. Lastly, having the gold values for the corresponding keys without the grounding passages cannot beat the performance with the original inputs. In particular, although the model can recover more information (i.e. better recall), not being grounded causes it to generate less consistent information (i.e. lower precision). When accompanied with groundings, the model achieves the best performance, emphasizing the importance of grounding.

5.7 Related Work

Natural Language Generation. Several data-to-text problems have been proposed with various input formats like Knowledge Graphs (Koncel-Kedziorski et al., 2019; Cheng et al., 2020), Abstract Meaning Representations (Flanigan et al., 2016; Ribeiro et al., 2019), tables and tree structured semantic frames (Bao et al., 2018; Chen et al., 2020; Parikh et al., 2020; Chen et al., 2021; Nan et al., 2021), and Resource Description Framework (Gardent

et al., 2017).

Towards a more controlled generation task, ToTTo (Parikh et al., 2020) was introduced for an open-domain table-to-text generation where only some of the cells are selected as the input. However, ToTTo and most existing datasets such as WikiBio (Lebret et al., 2016) and LogicNLG (Chen et al., 2020) focus on generating single sentences. Although generating long-form text is becoming a new frontier for NLP research (Roy et al., 2021b; Brahman et al., 2021), not many datasets and tasks have been proposed to explore this direction. Available datasets such as RoToWire (Wiseman et al., 2017) or MLB (Puduppully et al., 2019) are either small-scale or on single domain (e.g., Sports). Unlike prior works, we propose a controllable long-form entity description generation task that covers multiple domains and categories, including people, location, organization, event, etc.

Recently, Chen et al. (2021) presented the WikiTableT dataset for long-form text generation from multiple tables and meta data. In a more natural setting, our EntDeGen dataset uses factual and topical keys as guidance but still leaves a considerable amount of content selection from grounding passages to be done by the model. Our proposed grounded keys-to-text generation framework is generic and can easily be applied to other scenarios where the candidate set of grounding passages can be obtained via a simple internet search. This enable having access to the most up-to-date knowledge.

Factual Consistency Evaluation. Evaluating factual consistency of machine-generated outputs has gained growing attention in recent years. New approaches have been proposed mainly for tasks like abstractive summarization and machine translation (Zhang et al., 2020b; Sellam et al., 2020; Durmus et al., 2020). Some of these metrics are QA based and have been used to measure common information between documents/reference and summaries (Eyal et al., 2019; Scialom et al., 2019; Wang et al., 2020). Our proposed metric, MAFE, is inspired by these works.

5.8 Summary

This chapter studied entities as narrative elements and proposed a practical controllable task of generating narratives about entities grounded in world knowledge. We construct a large-scale dataset EntDeGen to facilitate research on this task. We proposed ranking-based models and an evaluation metric to gauge the factual correctness of generated narratives. Experiments showed the effectiveness of the proposed rankers to fetch relevant information required to generate a factual narrative. Human evaluations showed that EntDeGen poses a challenge to state-of-the-art models in terms of achieving human-level factuality in long-form generation. Our proposed dataset and task can also foster further research in the recently emerging retrieval augmented generations models (Lewis et al., 2020b; Zhang et al., 2021; Shuster et al., 2021) – where the retriever and generator components are trained end-to-end.

Chapter 6

Conclusion

In this chapter, we summarize our contributions and conclude with a discussion on future challenges and directions.

6.1 Contributions

In this dissertation, we investigated methods for advancing narrative *understanding* and *generation* by modeling key *narrative elements* that contribute to a good story. We designed new tasks and resources as well as modeling frameworks for efficient incorporation of narrative elements into the understanding and generation process.

We begin by taking cue phrases such as actions, entities or topic words as narrative elements resembling a plot (outline). We explored the problem of interactive storytelling, which leverages human and computer collaboration for creative language generation. To address this task, we presented two content-inducing approaches that take user-provided inputs as the story progresses and effectively incorporate them in the generated text. This human-computer collaboration is a step towards building an AI agent which can communicate with and assist humans (e.g., writers, learners with disabilities, and children) in a real-world situation and in an effective, scalable way.

In the following chapter, we proposed an emotion-aware story generation task for modeling the evolution of emotions in stories. To this end, we designed two emotion-

consistency rewards using a commonsense transformer (Bosselut et al., 2019) and an emotion classifier to regularize the story generation through reinforcement learning. Experiments demonstrated that our approach improved both content quality and emotion faithfulness of the generated stories. However, the various assumptions and choices made in this work and the specific characteristics of the dataset we chose can introduce biases and errors. For example, our approach uses COMET, a discourse-agnostic model, for separately extracting emotional reactions for each sentence. This may fail to maintain emotional consistency with the rest of the narrative. Such sources of errors and biases need further systematic investigation. Moreover, future models can use and improve upon contextualized discourse-aware commonsense inferences for extracting emotional reactions (Gabriel et al., 2021).

Next, we studied character(s) as key elements necessary for understanding narrative. To bridge the gap between human and machine character-centric narrative understanding, we introduced a new dataset containing about 9.5k literature summaries paired with descriptions of characters that appear in them. Using this dataset, we proposed tasks, namely *Character Identification* and *Character Description Generation* grounded on narrative summaries to investigate neural models' ability in understanding the narrative from the perspective of characters. Through several experiments and thorough analyses, we demonstrated that current state-of-the-art models are not up to the task of narrative understanding on the character level and leave plenty of room for improvements.

Finally, under a broader definition, we use entities (and not just characters) as narrative elements and explored generating narratives about them grounded in world knowledge. We proposed a grounded keys-to-text generation framework, where the user controls the information content using *factual* and *topical* guiding keys. To enable real-world application, the pertinent information to generate factual narratives is then fetched from a set of groundings (knowledge sources) using ranking-based models. We constructed a large-scale dataset EntDeGen to facilitate research on this task and presented an evaluation metric to assess the factual correctness. While proposed ranking strategies proved effective in providing relevant information, human evaluations showed

that models still struggle to achieve human-level factuality when generating longer text.

6.2 Future Directions

While this dissertation explored several approaches, our investigations shed light on several shortcomings and interesting future directions. Below, we discuss a few possible future directions:

Modeling Narrative Elements and their Evolution In this dissertation, we took several approaches towards modeling some of the narrative elements, which future work could explore further by explicitly modeling other narrative elements during generation. For example, modeling characters' roles, goals, and relationships with others and how they evolve and drive the story forward as well as the setting and theme of the story can be plausible directions. Moreover, we primarily focused on generating short stories. Moving to longer stories, we need to revise a lot of assumptions and focus more on paragraph-level modeling rather than a sentence/segment-level analysis. Inspired by how human writers keep track of the relations between entities and events, one can equip generators with a reader model that mentalizes and reasons about events and entities with explicit memory that gets updated as the story progresses.

Besides the development of the plot and characters, a good story should be engaging and have creative language. Reversals are surprises (change in directions) that affect storylines and give stories complexity. Understanding reversals and incorporating them while automatically generating stories enhance the stories' interestingness. Including figurative language and involving the reader via descriptive language (triggering five senses) are other fundamental tools that can be leveraged and incorporated into generation models.

Character-centric and Socially-aware NLP Models Current NLP models make an oversimplifying assumption that language is about 'content', aka what is said.[1] However,

[1]This only works in information-driven tasks such as QA.

according to linguistic theories language is about *who* says what to *whom*, in what *context*, and for what *goal* (Grice, 1975; Halliday and tian MIM Matthiessen, 2013). Therefore, interpreting and modeling these social factors of language (and the way they are used to convey social meaning) are essential to achieve human-level competence (Hovy and Yang, 2021). For instance, the implicit meaning of a content should be interpreted by considering speaker's or receiver's identity and their relationship. Taking *"I see you have every season of Sex and the City on DVD"* as an example, it might imply different meaning depending on the relationship between speakers. If it's the parent of the person, it can imply: *"I don't approve you of watching this show, you should get rid of those."*, or if it's a friend, it implies: *"I wish I had those. Can I borrow them sometime?"*. Current NLU and NLG models lack such commonsense of sociolinguistic competence.

Towards addressing these issues, we need to keep distilling knowledge and expanding symbolic (social) knowledge bases that can be used by machines to develop meaningful character-centric representations. Narratives which are rich in social and general knowledge can provide a great resource for knowledge acquisition and knowledge base construction. While there have been many formalisms created focusing on inferential knowledge (Sap et al., 2019), social and moral norms (Forbes et al., 2020), and social biases (Sap et al., 2020), there are still other aspects of social worlds that models should be aware of. For example, AI systems should understand acceptable behavior in different interpersonal relationships. Creating resources containing this type of knowledge enables studying how different interpersonal relationships impact interactions between people. While high-quality human-authored knowledge bases are too expensive to scale, recent large pre-trained language models offers a promising direction on automatic knowledge base construction using few-shot or prompt-based methods (Brown et al., 2020; Liu et al., 2021). And finally, it is essential to design new tasks that require character-centric reasoning and Theory of Mind (ToM) (Baron-Cohen, 1997).[2]

[2]Theory of mind is the knowledge of epistemic mental states that humans use to describe, predict, and explain behavior.

Commonsense Knowledge Grounding Our experiments with several state-of-the-art natural generation models showed that they generate texts that are often self-contradictory when they are inconsistent w.r.t their previous context or inconsistent w.r.t to facts and commonsense knowledge. For example, in a persona-guided dialogue, when being a vegetarian is given as persona, generating an answer like *"I went to a steak house with my friend."* is contradicting the fact that vegetarians do not each meat and steak is a kind of meat. Another known issue is generating implicit toxic and biased language targeting particular demographics. Many such issues can be due to insufficient knowledge grounding and lack of commonsense. A promising direction could rely on recently emerging Retrieval Augmented Generation (RAG) techniques (Lewis et al., 2020b; Shuster et al., 2021) to enrich the generation models with 'background stories' which are related to the context. Another possible avenue for future research is to develop new and efficient ways to incorporate external knowledge sources into generation models.

Chapter A

Emotion-Aware Storytelling Supplementary

A.1 Training and Experiment Setup

A.1.1 Obtaining Emotion Reactions During RL-EM Training

During self-critical training, for each training instance, two stories are generated: $\boldsymbol{y}^s$ which is sampled from the model's probability distribution, and $\hat{\boldsymbol{y}}$ which is greedily decoded. Then for each sampled or greedy story, the value of the reward is computed. For computing the EC-EM reward, we need to identify and track the protagonist and obtain his/her emotional reactions for each sentence of the sampled or greedy story. This requires identifying the protagonist and determining his/her role, *Agent* or *Other*, in every sentence so that the appropriate argument for COMET (`xReact` or `oReact`) can be chosen. In principle, this can be done using the annotation pipeline described in §4.1 of the main paper. However, doing this is computationally prohibitive during training as the pipeline requires running dependency parsing for each sentence and coreference resolution for every sampled or greedy story. To this end, upon analyzing our corpus and some generated stories, we devised several heuristics that approximate the tasks (i.e. identifying protagonists and their roles) with high accuracy. We describe

these heuristics below.

For identifying the protagonist, we use the following heuristics. The first heuristic is based on the observation that if the narrator features in a story, the story primarily focuses on the narrator and his/her experiences, thus making him/her the protagonist. So our first heuristic is that if first person pronouns (I, We) appear in the story, they are considered the protagonist. Our second heuristic is based on the observation that the protagonist is usually introduced fairly early in the story. Especially, in our case, where the stories are 5-sentence long, the protagonist appears mostly in the first couple of sentences of the story. With this in mind, we define the first noun that appears in a lexicon of common protagonists as the protagonist of the story. This lexicon consists of terms for `Male_Char`, `Female_Char`, `Social_Group`, `Generic_People`, and the NLTK name corpus. Example of these terms are shown in Table A.1. This also let's us identify the gender of the protagonist (using the lexicon category that the protagonist belongs to) and hence the pronouns that will be used in the following sentences to refer to the protagonist. This combination of protagonist's mentions and pronouns lets us track him/her throughout the story.

For identifying the role of the protagonist, if the first noun that occurs in a sentence matches the protagonist or his/her corresponding pronoun, we assume that the protagonist's role is *Agent*, otherwise the role is *Other*. Depending on this role, we use `xReact` or `oReact` when obtaining protagonist's emotional reaction in that sentence using COMET.

`Male_Char`	husband, father, dad, dady, brother, grandpa, granddad, son, nephew, man, boy, boyfriend
`Female_Char`	wife, mother, mom, momy, sister, grandma, grandmom, niece, daughter, nana, woman, girl, girlfriend
`Social_Group`	family, parents, grandparents, children, kids, couple, friends, boys, girls, band
`Generic_People`	cousin, friend, fiance, boss, manager, assistant, doctor, nurse

Table A.1: Predefined terms used for tracking the protagonist.

A.1.2 Training Hyper-parameters

Our proposed models follow the setting of medium-sized GPT-2 Radford et al. (2019) (345 million parameters) that used a 24-layer decoder-only transformer, 1024-dimensional hidden states, and 16 attention heads. The stories are encoded using BPE with vocabulary size of $50,257$. We set the maximum sequence length to 128 tokens, as it is large enough to contain complete stories and additional inputs. We use Adam optimization Kingma and Ba (2014) with an initial learning rate of 10^{-5} and minibatch of size 4. For stability, we first pre-train the models with teacher forcing until convergence, then fine-tune them with the mixed loss. Hyper-parameter $\gamma = 0.97$ is tuned manually on the validation set. All models were trained until there was no improvement on the validation set performance. We use a NVIDIA GTX 1080 Ti GPU machine to train our models. At inference time, we generate stories using top-k sampling scheme Fan et al. (2018c) with k=40 and a softmax temperature of 0.7.

For generating commonsense inferences about the protagonist's emotions using COMET, we use greedy decoding algorithm since it has been shown to have superior performance as evaluated by humans Bosselut et al. (2019).

A.1.3 Evaluation Measures

In this section we provide details about the automatic and manual measures used to evaluate our models.

Automatic To compute Arc-word and Seg-word measures, we use NRC Affect Intensity Lexicon Mohammad (2018). This lexicon contains words with corresponding emotion-intensities for different *basic emotions*. To find emotionally expressive words in a given piece of text (e.g., a story segment), we create a dictionary of words with emotion intensity higher than 0.5 for each of our *basic emotions*.

Manual During the manual evaluation, we conducted pairwise comparison of the models on Amazon Mechanical Turk (AMT). To ensure high quality of evaluation, we selected turkers that had an approval rate greater than 97%, had at least $1,000$ approved

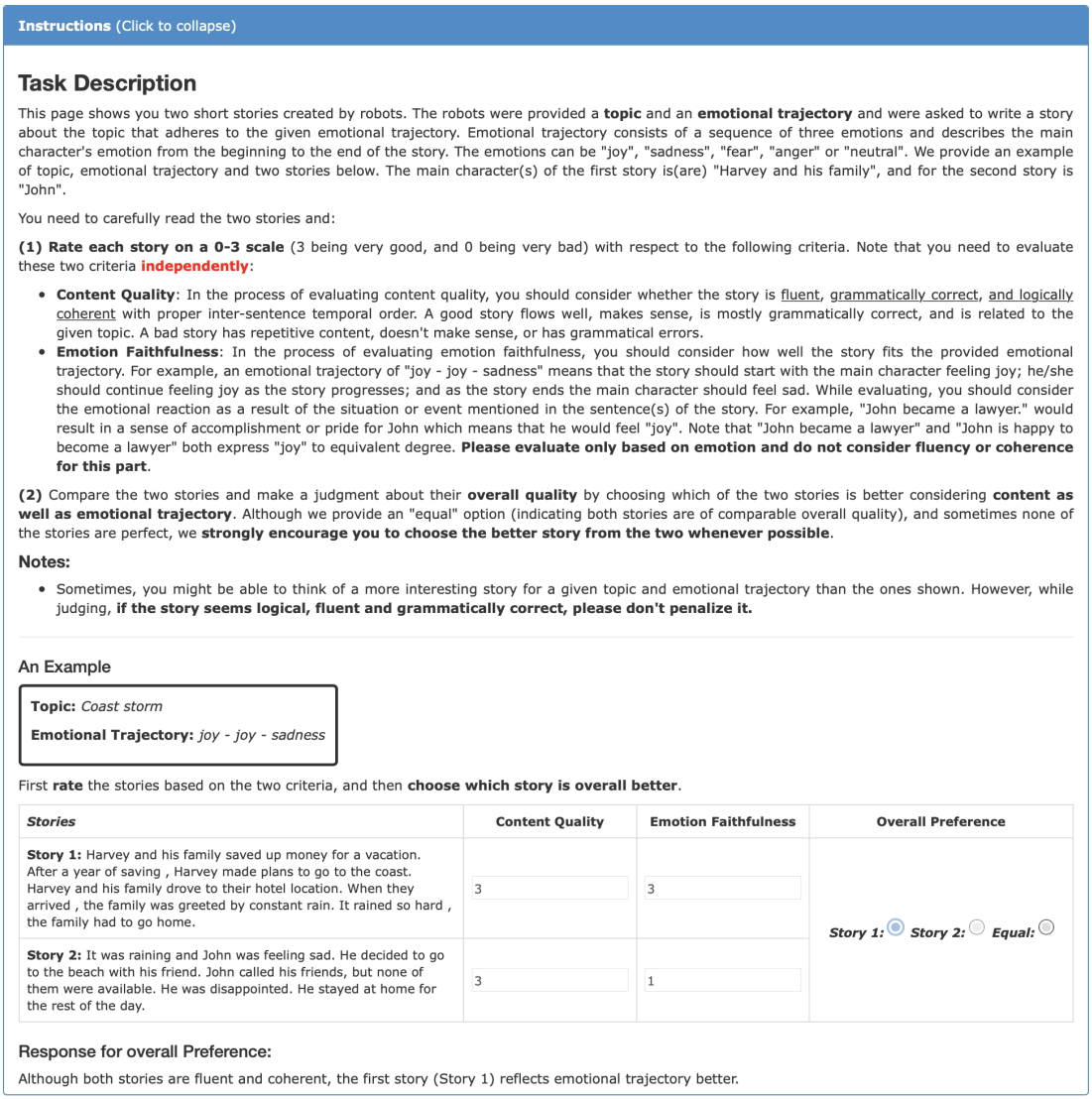


Figure A.1: An screenshot of manual evaluation on AMT.

HITS, and were located in the U.S. For each pairwise annotation, we showed the inputs (title and emotion arc) and two stories generated using the two models being compared. In order to avoid biases, we randomly shuffled the order in which the stories from the two models were shown to the turkers. We provided instructions to the turkers explaining the annotations and also provided examples. Following this process, each pair of stories was annotated by three turkers. Figure A.1 shows a screenshot of our setup on AMT.

Models	Domain	Accuracy (Jaccard)	Mico F1	Macro F1
Meisheri and Dey (2018)	tweets	0.582	0.694	0.534
Baziotis et al. (2018)	tweets	0.595	0.709	0.542
Kant et al. (2019)	tweets	0.577	0.690	0.561
BERT_{large} (ours)	tweets	0.595	0.708	0.522
BERT_{large} (ours)	stories	0.617	0.650	0.557

Table A.2: Emotion classification results on the tweets dataset (upper block), and the automatically annotated story corpus (lower block).

A.2 Supplementary Results

A.2.1 Emotion Classification

Our Ec-Clf reward captures the protagonist's emotions using our emotion classifier. In this section we provide details about its evaluation.

We first evaluate the classifier on the tweets corpus Mohammad et al. (2018)[1] by comparing it with several strong baselines Kant et al. (2019). For this comparison, we trained all models on the training set of the corpora and tested them on a held-out test set. The models were evaluated using Jaccard Index based accuracy, and Micro and Macro F1 scores. This evaluation set-up (train-validation-test splits and choice of evaluation metrics) is as suggested in the challenge that provided the corpus (SemEval Task1:E-c challenge). The results of this comparison is shown in the top half of Table A.2. We can see that our emotion classifier, BERT_{large}, is superior or competitive with other models.

The results reported above show that the model performs well for emotion classification in tweets. However, our goal is to design a model that can be used to track protagonist's emotions in stories. As described in the main paper, we further fine-tuned this classifier on our automatically annotated story corpora (described in the paper in §3.3.1). We also evaluated the classifier on a held-out portion of this story corpora consisting of about $1,201$ stories ($6,005$ sentences in total). The results are reported in the last row of Table A.2. The classifier achieves a (Jaccard Index) accuracy of 61.75% and micro and macro F1 scores of 0.650 and 0.557 respectively. Note that this is different

[1] https://competitions.codalab.org/competitions/17751

from the evaluation reported in the paper, which was conducted on a subset of stories annotated by humans.

A.2.2 Manual Annotation for Protagonist's Emotions

As described in the paper (§3.2.1), the emotion classifier was also evaluated on a subset of 50 randomly selected stories (250 sentences) manually annotated for the emotions experienced by their protagonists. This annotation was done on Amazon Mechanical Turk. To ensure good quality of the annotations, we selected turkers who had an approval rate greater than 97%, had at least 1,000 approved HITS, and were located in the U.S. The Fleiss kappa (inter-annotator agreement) was also moderate ($\kappa = 0.55$). We also analyzed the annotations to identify major sources of disagreements between the turkers. We found that most disagreements occurred between *neutral* and *joy*; and also between *sadness* and *anger*. The overall label distribution for this human annotated set is shown in Figure A.2.

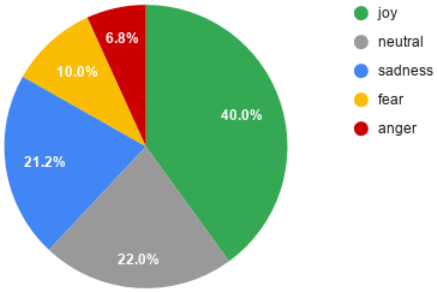


Figure A.2: Label distribution of human annotated set.

A.2.3 Emotional-Aware Storytelling

We also compare various models on the most common emotion arcs in our corpus. Figure A.3 shows the Arc-acc of various models on the 10 most common arcs. We can see that all models perform very well on "joy → joy → joy" as compared to other emotion arcs. This is because this is the most common emotion arc (34% of the training data) in our corpus, which results in availability of significant number of training examples for this arc. Nevertheless, for all arcs, RL-CLF consistently outperforms all other models indicating a better control over the desired emotion arc.

Supplementary qualitative examples We provide more qualitative examples in Figure A.4. In the figure we show stories generated by our model for a given title and

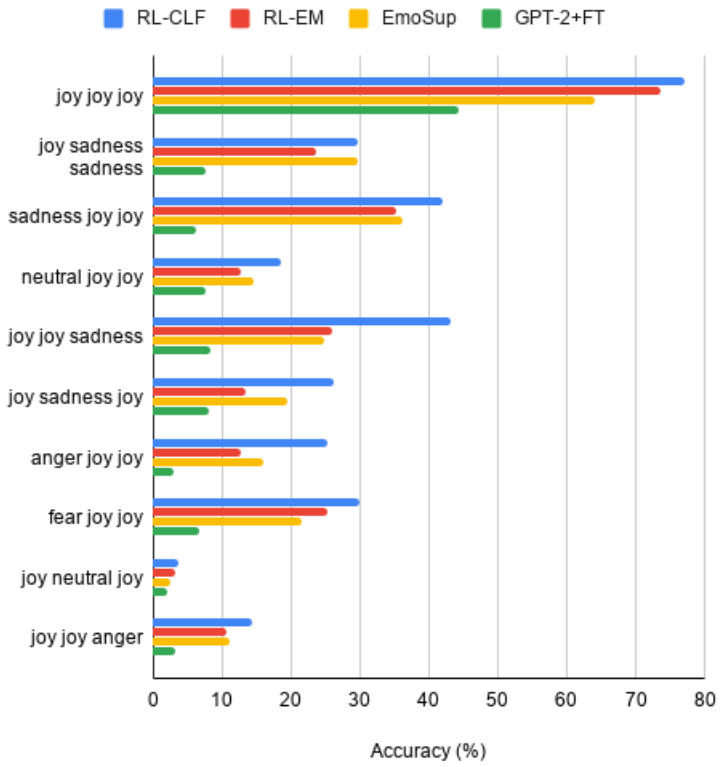


Figure A.3: Arc-acc of various models on the 10 most common arcs in our corpus. RL-CLF outperforms other models for almost all arcs.

emotion arc for the protagonist.

Title: A new outfit
Emotion arc: joy → joy → joy
Story: Sara went to the mall. She wanted to find something cute to wear. She noticed a beautiful red dress she liked. Sara bought the dress. She was happy that she found something nice.

Title: Wedding
Emotion arc: joy → anger → joy
Story: Ben is getting married. He is having fun. Suddenly , Ben gets into a huge fight with a guy. Ben gets out of the wedding early. Ben is so happy to have his wedding over.

Title: Study
Emotion arc: fear → fear → sadness
Story: Tina was so nervous for her big exam. She did not study at all. She was so afraid she would fail. She ended up failing. Tina felt very sad.

Title: The big storm
Emotion arc: joy → joy → sadness
Story: It was a beautiful day in the city. The sky was clear and the weather was mild. The school was hosting an evening celebration. A big storm rolled through and the students were worried. The storm destroyed all the decorations and ruined the party.

Title: Delayed flight
Emotion arc: joy → anger → sadness
Story: Tom was going on a vacation. He decided to go to Florida. Unfortunately his flight was delayed. He was so frustrated he called his airline. His airline cancelled his flight.

Title: The new pet
Emotion arc: neutral → joy → joy
Story: Sam was walking around the neighborhood. She saw a cute little dog. She decided to take him home. He got along well with everyone. Sam was glad to have a companion.

Title: Larry practice yoga
Emotion arc: fear → joy → joy
Story: Larry has always felt nervous about yoga. He has tried many times to practice but has never gotten the hang of it. He decides to take a yoga class at his local yoga studio. He is amazed by the benefits and feels confident about his yoga practice. Larry is happy he learned to enjoy yoga.

Figure A.4: Qualitative examples of generated stories given a title and an emotion arc.

Chapter B

LiSCU Supplementary

B.1 Collecting Annotations from Crowd Workers

To alleviate the limitations of crowd-sourcing and ensure high quality of annotations, we took several steps. First, we conducted a pilot annotation exercise where we (authors) assessed the feasibility of the proposed task on a subset (250 instances) of the data. This pilot annotation helped us set up the task on AMT in a way that would make the task feasible for turkers (e.g. by asking clear concise questions). Second, we designed our setup to avoid annotator fatigue by asking them to read the summary context once and answer questions about all characters in that summary. Third, we ran a few experiments on AMT (before annotating the entire set) where we also included a 'comment' section for turkers to allow them to bring up issues or ambiguities in our setup. We then manually analyzed the results and modified the tasks based on the comments. Finally, after annotating the entire set, we computed inter-annotator agreement as a way to ensure trust in the annotation quality. We found reasonable agreements between annotators as reported in Footnote 10 of the paper. We would also like to mention that we received several comments from the annotators that they found the task very interesting and enjoyable.

Model	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	BERT-F1
Length Truncated Input					
GPT2-L	0.90/0.71/0.60/0.59	20.33/19.94/18.81/19.39	4.34/3.60/3.36/3.32	18.50/17.91/17.14/17.84	76.23/80.11/76.53/75.81
Longformer	1.86/1.06/0.92/0.70	24.16/21.78/20.55/20.20	6.80/4.62/4.33/3.98	22.05/19.65/18.71/18.26	85.60/84.92/84.46/84.24
Coreference Truncated Input					
GPT2-L	0.82/0.63/0.53/0.58	19.80/19.24/17.96/18.49	3.86/3.24/3.06/3.04	17.79/17.39/16.46/16.62	76.16/79.52/78.33/80.23
Longformer	1.78/1.09/0.77/0.65	23.32/22.23/20.23/19.90	6.04/4.80/3.96/3.57	21.41/20.22/18.16/17.70	85.43/85.07/84.47/84.12
Full Length Input					
Longformer	2.15/1.31/1.04/0.65	24.47/22.56/20.90/20.63	6.98/5.37/4.63/3.84	21.91/20.40/18.85/18.48	85.66/85.11/84.57/84.35

Table B.1: Breakdown results on subsets of test set with annotated fact coverage as all/most/some/little.

Generated Description	Comments
Book title: The Three Sisters **Character name:** Vershinin **Summary:** Three Sisters mainly follows the story of–wait for it–three sisters: Olga, Masha, and Irina Prozorov. They live with their brother, Andrey, in a big house on the edge of a small Russian town. The townspeople are kinda backward and boring compared to their educated and culture-lovin' family, so this set of sibs is not too fond of the town to begin with. Believe it or not, the only halfway interesting people around are the guys in the military. Basically, the Prozorov kids are worldly, well-educated army brats. And being in the army in Tsarist Russia pretty much meant you were in with the aristocracy and, once you got through the fighting stuff, probably developed a taste for the finer things in life. So ever since the family moved from Moscow eleven years prior (with their father, now dead), the sisters have obsessed over the dream of moving back to the big city. But guess what? It's not happening. Olga, the eldest, is a spinster schoolteacher and eventually becomes a headmistress living with her elderly maid. Masha, the middle sister, is married to another schoolteacher, Kulygin, whom she despises for his small-mindedness. She has an affair with the officer Vershinin because he's given to just the sort of philosophizing that really starts her engines. And guess what happens: the affair ends in heartbreak. Poor Irina, the youngest, has fanciful ideas about the value of work, but soon realizes that, in reality, work sucks the life out of her. She's also in love with the idea of love, but doesn't get to experience it. Finally she comes around to saying "yes" to Baron Tuzenbach, a friendly but ugly man who's been after her for years. On the day they're leaving to get married, he gets shot in a duel. Bummer. Andrey, the brother, gives up his intellectual dreams to pursue a town girl, Natasha. They marry, have kids, and little by little she takes over the estate. At the end of the play, the upper-class Prozorovs are pretty much evicted from their own house, while Natasha, a symbol of the working class, is on the rise. Allegory, much? For anyone not already reaching for the history books, this was just a few years before the Bolshevik Revolution of 1917–and you better believe smart Ruskies like Chekhov could already sense some storms on the horizon. **Generated Character Description:** Vershinin is one of the most interesting characters in Three Sisters. He's a young officer in the Russian army, and he's got a lot of time on his hands. But that doesn't stop him from having an affair with Irina, the youngest sister, who falls in love with him because she's attracted to his philosophical ideas about the meaning of life.	According to the summary "Masha", the middle sister, is the one who has an affair with the officer Vershinin, and not the youngest sister, Irina.

Table B.2: Qualitative example 1 for the generated descriptions. Words in red correspond to hallucinated or missing content, words in green correspond to faithful information.

Generated Description	Comments
Book title: The Sun Also Rises **Character name:** Pedro Romero **Summary**: Jake Barnes and his expatriate friends live in the topsy-turvy, hedonistic (sensual and self-indulgent) world of post-World War I Paris. There, they occasionally work, but spend most of their time partying, drinking, and arguing. From Jake's perspective, we meet the cast of characters that populates his story: the most important among them are Robert Cohn, a weak-willed, down-on-his-luck Princeton grad and unsuccessful writer, and Lady Brett Ashley, an exciting, beautiful, and unpredictable British divorcee. Although Jake and Brett are actually in love, they aren't together, presumably because a mysterious war wound has rendered Jake impotent. Cohn falls in love with Brett (as everyone does) and, despite the fact that she's not terribly impressed with him, she secretly goes on a trip with him to the Spanish resort town of San Sebastian. Cohn is infatuated with Brett–he's completely smitten. We're talking truly, madly, deeply in smit. Unfortunately for Cohn (and for everyone, for that matter), Brett is engaged to a wealthy, charming, and utterly inept drunkard named Mike. Jake's whimsical friend Bill returns to Paris from a trip and a plan is born: everyone agrees to decamp to Spain for some fishing and the running of the bulls in Pamplona. On their brief fishing trip, Bill and Jake have a splendid time communing with nature and with each other, but the relaxation quickly comes to an end. They return to civilization and meet up with Brett, Mike, and Cohn in Pamplona for a weeklong orgy of bullfights, alcohol, and high drama. Jake has a true passion (aficion) for bullfighting, but everyone else is simply there to have a good time. Brett begins a scandalous affair with a passionate and talented young bull-fighter, Pedro Romero. Jake feels terrible for many reasons–among them is the fear that he has corrupted Romero in some way by introducing him to Brett. Cohn's thwarted infatuation with Brett leads to arguments with everyone and, finally, he beats the unfortunate Romero to a bloody pulp. As the fiesta winds down, everyone leaves Pamplona in various states of anxiety, depression and frustration.Jake heads to San Sebastian, where he intends to decompress alone for a while. Unfortunately, desperate telegrams from Brett arrive immediately. He goes to her in Madrid, where she is alone, having sent Romero away. For the first time, we see Brett truly vulnerable, afraid, and guilty. The future looks just as bleak–Jake and Brett agree again that, even though they love each other, they can't be together. **Generated Character Description:** Pedro Romero is a young Spanish bull-fighter who is in love with Lady Brett Ashley, a beautiful British divorcee. He's a talented and talented young man, and, despite the fact that he is only twenty-five years old, he has the potential to be one of the best bullfighters in Pamplona. Jake feels terrible for introducing him to Brett, fearing that it has corrupted him.	The description captures most of the important details about the character.

Table B.3: Qualitative example 2 for the generated descriptions. Words in red correspond to hallucinated or missing content, and words in green correspond to faithful information.

Generated Description	Comments
Book title: The Waves **Character name:** Neville **Summary:** The story begins by introducing us to the novel's six (yup, you read that right) narrators, Bernard, Neville, Louis, Jinny, Susan, and Rhoda, who meet as children in a nursery. During this phase of the novel, we learn a lot about the characters' personalities and their relationships to each other. After looking on as our new friends get embroiled in some kid-level dramas (e.g., trouble in math class and unrequited crushes), the six children head off to their respective boarding schools. At that time, the boys meet Percival, whom everyone seems to revere (and Neville falls in love with). The protagonists then all graduate and proceed into their adult careers (with a stop at university along the way, for some). At some point in there, Percival becomes friends with the girls as well, though we're not sure when that actually occurs. The narrators' paths diverge quite a bit as the novel progresses. After enduring a stint in a Swiss school, Susan returns to her beloved hometown, gets married, and starts having babies. Meanwhile, Bernard apparently moves to Waterloo (that's not entirely clear, but Woolf drops some clues to that effect), and we're not entirely sure what he does there, other than shave and make up stories about pedestrians. Meanwhile, Rhoda, Louis, Neville, and Jinny go to live in London, and their life paths are all over the map: Louis works for a shipping company, Jinny is a socialite, and Neville is a classics professor (we don't learn Rhoda's profession). Midway through the book, the friends meet up for dinner in London to see Percival off before he leaves to work in India, at which point Bernard announces that he is getting married. An unspecified amount of time later, Percival falls from his horse in India and dies, and our narrators are devastated. The death occurs just as Bernard's son is born, creating some serious cognitive dissonance for poor Bernard. The characters then struggle with aging and reflect upon the progress of their lives and adulthood. Toward the end, they all meet up at Hampton Court and experience the aches and pains that come when old friends reconnect after a long time has passed. The novel ends with Bernard talking to an apparent stranger, attempting to sum up the lives of the six narrators (i.e., the events of the novel) and work out his philosophies of language and lifeand that's why the last chapter is fifty pages long. He reflects upon his lifelong struggle to turn his phrasemaking into something meaningful and, despite the major ups and downs he's experienced along the way, he resolves to keep on trying. **Generated Character Description:** Neville is one of the six narrators. He's a classics professor at Oxford University and, like the other five, he's also a classicist. We don't learn a whole lot about Neville, other than the fact that he falls in love with Percival, which makes him a bit of an oddball. In fact, we're not even sure what he does in his spare time away from Oxford.	The model hallucinated some details such as being a professor at Oxford University while the summary only mentions: He lives in London and is a classics professor. Also, phrases such as "We don't learn a whole lot about" appears quite frequently in the generated descriptions.

Table B.4: Qualitative example 3 for the generated descriptions. Words in red correspond to hallucinated or missing content, words in green correspond to faithful information, and underline corresponds to generic repetitive content.

Generated Description	Comments
Book title: Travels with Charley **Character name:** Charley the Dog **Summary:** Because he's feeling pretty out of touch with his own country–and he's considered a great American author and all that–John Steinbeck decides to take a road trip around the U.S. to check it out and get a sense of where Americans and their hometowns are at in 1960. To get all prepped, he commissions a souped-up truck with a little house on the back that he can live in when he isn't crashing at hotels. He calls the truck "Rocinante" after Don Quixote's horse–clever, huh? When he's all set (and after a small run-in with a hurricane just before he was supposed to leave), he and Charley (his French poodle) hit the road. He starts out by driving over into Connecticut from his home in Long Island (with some assists from ferries, natch) and then heads north into New England. Along the way, he meets a pretty colorful group of characters and learns about their ways of life and their perspectives on the country and its politics. Also, he kind of takes the temperature of regional "temperaments" along the way. Then he comes back down out of New England and heads west, crossing through New York. He tries to cut through Canada, but he gets into a kerfuffle at the border because Charley doesn't have his proof of rabies vaccination, so he has to turn around. Steinbeck then passes through the Midwest, continuing to offer his reflections and thoughts about the people and places he encounters along the way. When he gets to Chicago, he puts Charley in a kennel and enjoys a couple of days with his wife, who flew out to meet him. He doesn't give us details of their time together, though. After that brief interlude, he heads further west into Minnesota and Wisconsin. He hits bad traffic and gets lost around the Twin Cities, and he's charmed by Wisconsin and its dells. He also visits Sauk Centre, the birthplace of author Sinclair Lewis. [...] Then, he heads to Washington. Along the way, Charley ends up getting sick with a bladder problem, and Steinbeck tries to get him some help from a vet in Spokane. The doc isn't actually super-helpful, though. Still, they get the problem resolved enough that they can continue driving. Heading to Seattle, Steinbeck finds the place changed a lot from the last time he was there (and not for the better, sadly). Then he heads into Oregon, where his tire blows out. Through the kindness of a dude at the one service station open in that area on a Sunday, though, he gets some new heavy-duty tires and is on his way pretty quickly. Then he heads south through California, visiting Salinas (his hometown) on the way. [...] After that little break, Steinbeck goes to Louisiana so he can see for himself what's going on there in terms of racial tensions (which appear to be pretty high). There was a well-known protest going on: two African American children had been admitted to a New Orleans school, and lots of people had a huge problem with that fact. So, some women known as the Cheerleaders basically stood outside every morning to scream at these poor children as they headed into school. Steinbeck went to get a firsthand look at this spectacle and was, as you might imagine, disgusted. He then headed back home, finally deciding he was "done" with the journey somewhere in Virginia. **Generated Character Description:** When we first meet Charley, he's a French poodle. He's Steinbeck's pet and his companion for most of the road trip, and we get to see him grow up a bit. In fact, we see a lot of him in the book, from the time he first meets him until he gets to Chicago. We also get a glimpse of his personality as a character in his own right, when he tells us, "He was a good-natured, loving dog" (1.2.1).	The description never mention anything about the Charley's health struggles as appeared in the summary we have: "Along the way, Charley ends up getting sick with a bladder problem, and Steinbeck tries to get him some help from a vet in Spokane. The doc isn't actually super-helpful, though."

Table B.5: Qualitative example 4 for the generated descriptions. words in green correspond to faithful information, and underline corresponds to generic repetitive content.

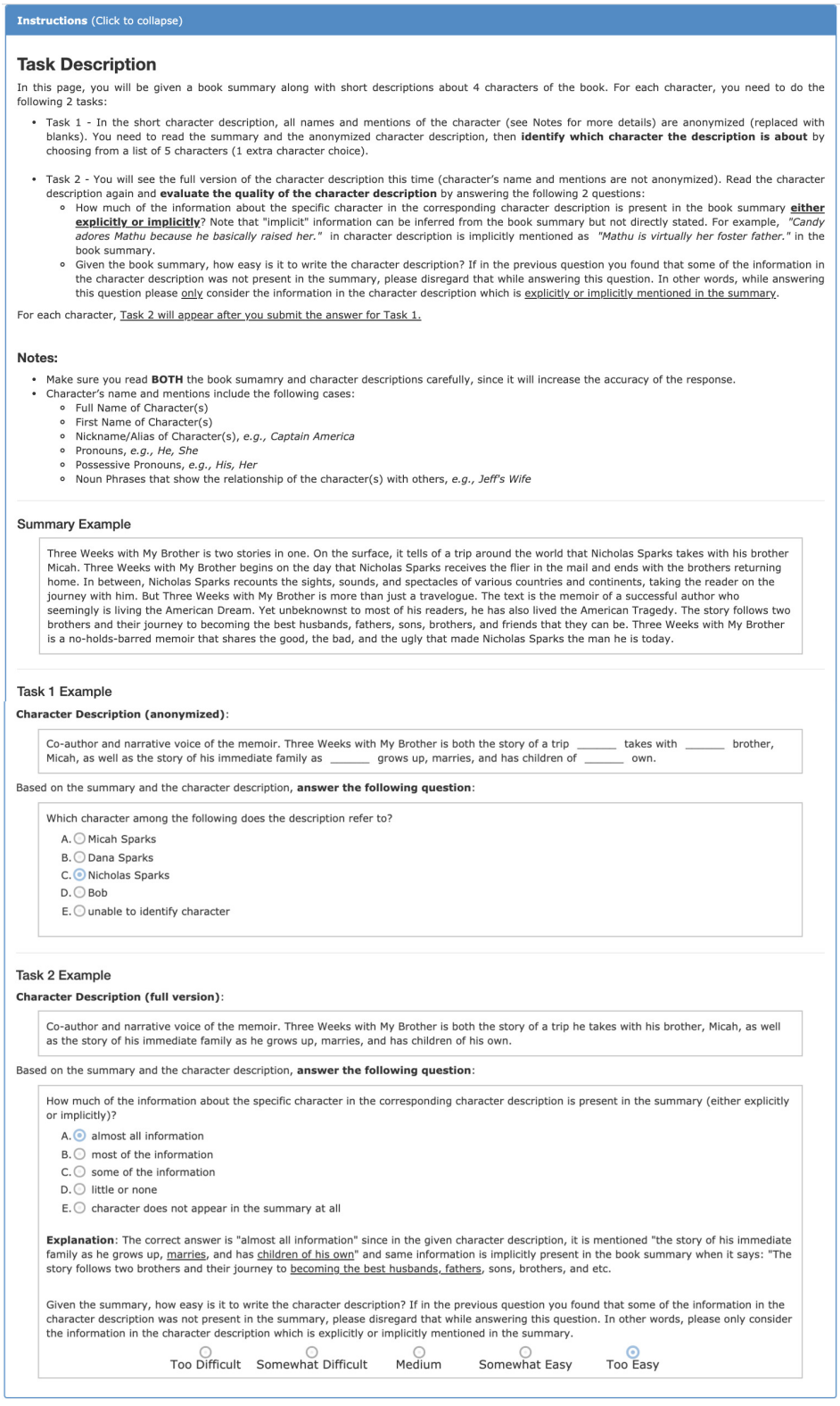


Figure B.1: An illustration of human assessment on AMT.

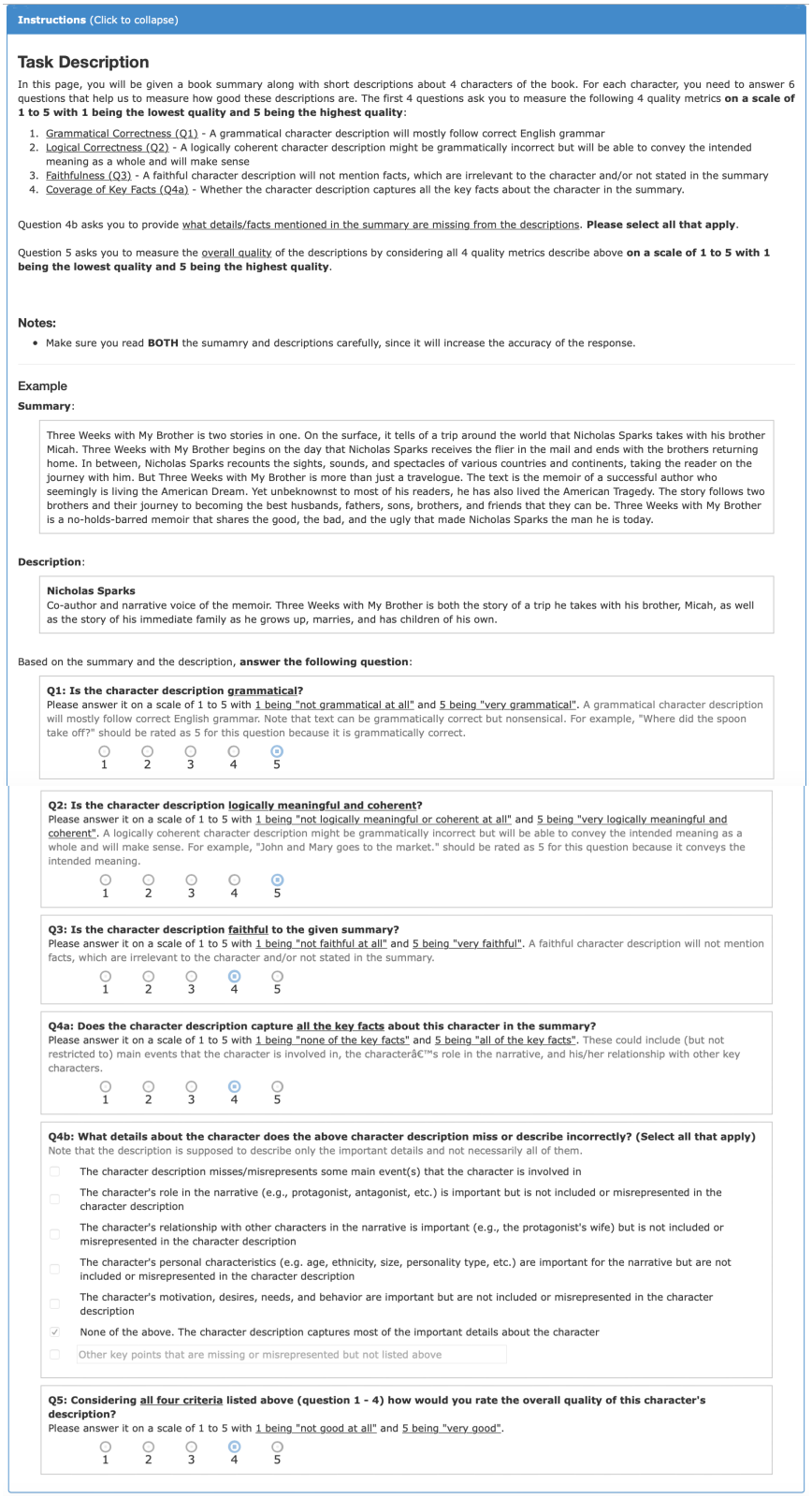


Figure B.2: An illustration of human evaluation for generated character description.

Chapter C

ENTDESCRIPTOR Supplementary

C.1 Implementation Details

Baselines. Top-10 grounding passages were used to train and test all baselines. We use the Transformer library Wolf et al. (2019a). Each baseline was trained for 3 epochs with effective batch size of 8, and initial learning rate of 5e-6 for T5 and BART, and 1e-4 for PEGASUS. We use the maximum input length of 512 tokens. During inference, we use beam search decoding with 5 beams, and repetition penalty of 1.2. Note that we use the BART-L model finetuned on XSUM dataset as our initial weights. Similarly, we use google's PEGASUS model finetuned on XSUM. The experiments are conducted in PyTorch framework using Quadro RTX 6000 GPU.

Rankers. The contrastive dense ranker was trained for 10 epochs with 2e-4 learning rate. The autoregressive ranker was trained for total of 30,000 steps with learning rate and weight decay of 1e-5 and 0.01, respectively. Rankers were trained using 4x Nvidia V100 GPU machines, each with 32G memory.

Question Generation in MAFE. The question generation module (QG) in MAFE evaluation metric, generates questions using beam search decoding with 10 beams.

Model	Ranker	R-1	R-2	R-L
Extractive-QA	n/a	22.37	6.28	19.85
T5-Large	Unranked	28.74	10.31	26.8
	Tf-idf	29.49	10.54	26.70
	Contrastive Dense	30.92	12.19	28.14
	Autoregressive	31.45	12.59	28.64
	Rouge2 (oracle)	33.75	14.84	30.84
BART-L	Unranked	33.52	14.39	30.82
	Tf-idf	34.00	14.54	31.14
	Contrastive Dense	35.26	16.04	32.46
	Autoregressive	**35.80**	**16.62**	**32.97**
	Rouge2 (oracle)	37.72	18.57	34.87
PEGASUS	Unranked	29.23	12.46	27.11
	Tf-idf	29.43	12.38	27.25
	Contrastive Dense	31.21	14.16	28.94
	Autoregressive	32.03	14.90	29.75
	Rouge2 (oracle)	34.01	17.03	31.71

Table C.1: Results (ROUGE; Lin (2004)).

C.2 Experimental Results

C.2.1 Automatic Evaluation

We report all ROUGE-1 (unigram overlap), ROUGE-2 (bigram overlap) and ROUGE-L (longest matching sequence) scores in Table C.1.

C.2.2 Human Evaluations

We restricted the pool of workers to those who were located in the US, or CA, and had a 95% approval rate for at least $1,000$ previous annotations. We use a pay rate of $15 per hour approximately based on our estimation of time needed to complete the task.

We depict our annotation layouts for evaluating precision-oriented and recall-oriented (both w.r.t reference and factual triples) factuality in Figure C.2, C.3, and C.4. Likert scale of 1-5 and binary scores (supported/not supported) are used when evaluating recall w.r.t references, and factual triples, respectively. These scores are then normalized

Figure C.1: ENTDEGEN Entity Domain Distribution of Correlation Analysis Subset.

and averaged to obtain the final recall-oriented score.

Figure C.2: An illustration of human evaluation of precision-oriented factuality. Generated paragraphs are presented one sentence at a time and are evaluated on how well they are supported by the references.

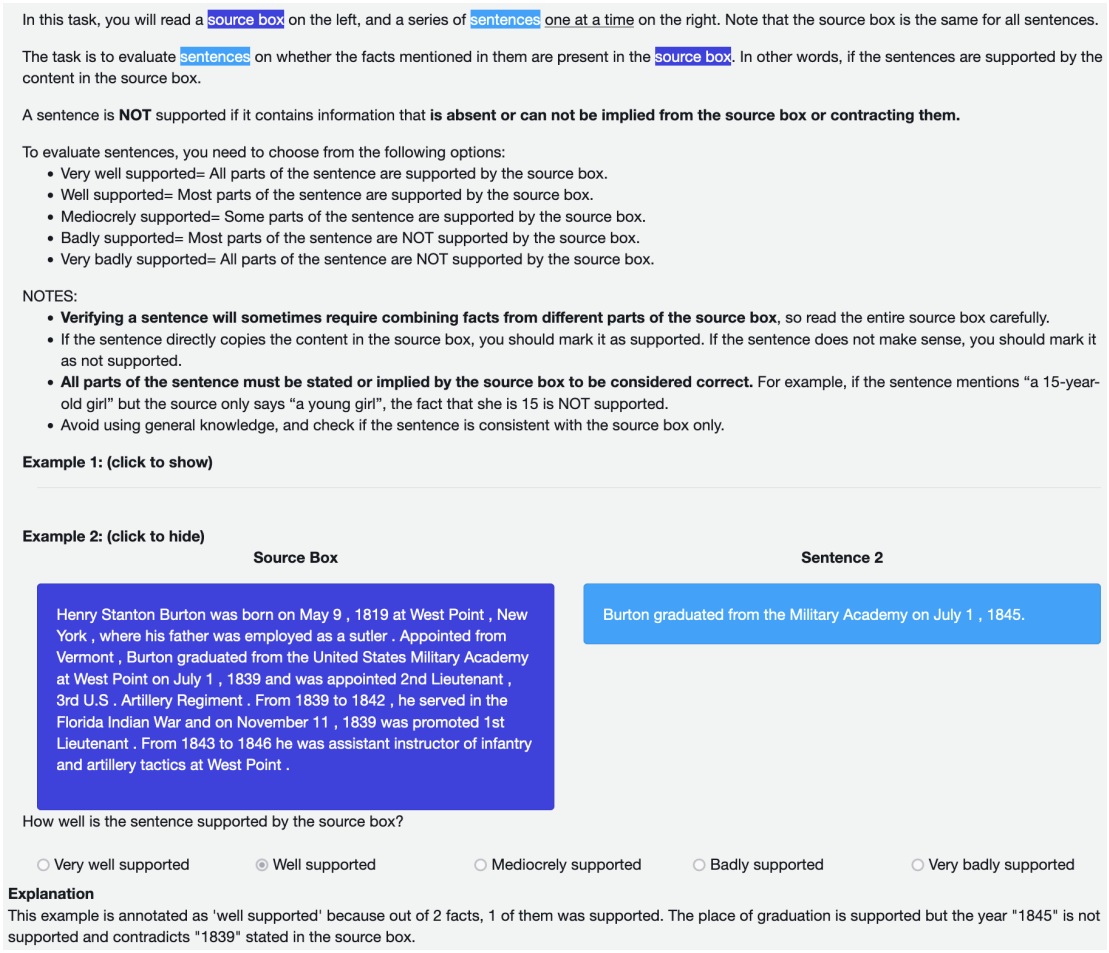


Figure C.3: An illustration of human evaluation of recall-oriented factuality w.r.t reference. References are presented one sentence at a time and are evaluated on how well they are supported by the generated paragraphs.

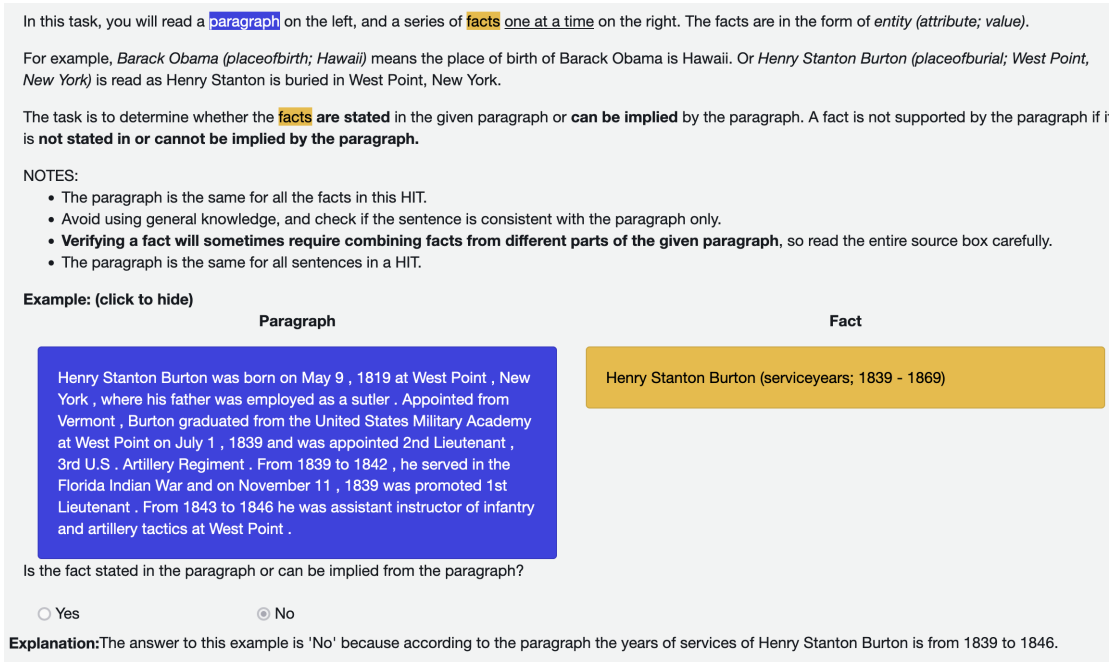


Figure C.4: An illustration of human evaluation of recall-oriented factuality w.r.t factual triples. Factual triples are presented one at a time and are evaluated on whether they are supported by the generated paragraphs or not.

Bibliography

Apoorv Agarwal, Sriramkumar Balasubramanian, Anup Kotalwar, Jiehan Zheng, and Owen Rambow. 2014. Frame semantic tree kernels for social network extraction from text. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 211–219, Gothenburg, Sweden. Association for Computational Linguistics.

Prithviraj Ammanabrolu, Ethan Tien, Wesley Cheung, Zhaochen Luo, William Ma, Lara J. Martin, and Mark O. Riedl. 2020. Story realization: Expanding plot events into sentences. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 7375–7382.

Reinald Kim Amplayo, Seonjae Lim, and Seung-won Hwang. 2018. Entity commonsense representation for neural abstractive summarization. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 697–707, New Orleans, Louisiana. Association for Computational Linguistics.

Tory S. Anderson. 2015. Goal reasoning and narrative cognition. In *Annual Conference on Advances in Cognitive Systems: Workshop on Goal Reasoning*.

Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

David Bamman, Brendan O'Connor, and Noah A. Smith. 2013a. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 352–361, Sofia, Bulgaria. Association for Computational Linguistics.

David Bamman, Brendan O'Connor, and Noah A. Smith. 2013b. Learning latent personas of film characters. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 352–361, Sofia, Bulgaria. Association for Computational Linguistics.

David Bamman, Ted Underwood, and Noah A. Smith. 2014a. A bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 370–379, Baltimore, Maryland. Association for Computational Linguistics.

David Bamman, Ted Underwood, and Noah A. Smith. 2014b. A Bayesian mixed effects model of literary character. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 370–379, Baltimore, Maryland. Association for Computational Linguistics.

Junwei Bao, Duyu Tang, Nan Duan, Zhao Yan, Yuanhua Lv, M. Zhou, and T. Zhao. 2018. Table-to-text: Describing table region with natural language. In *AAAI*.

Simon Baron-Cohen. 1997. *Mindblindness: An essay on autism and theory of mind*. MIT Press.

Somnath Basu Roy Chowdhury, Faeze Brahman, and Snigdha Chaturvedi. 2021. Is everything in order? a simple way to order sentences. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10769–10779, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Christos Baziotis, Athanasiou Nikolaos, Alexandra Chronopoulou, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, Shrikanth S. Narayanan, and Alexandros Potamianos. 2018. NTUA-SLP at semeval-2018 task 1: Predicting affective content in tweets with deep attentive RNNs and transfer learning. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 245–255.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *ArXiv*, abs/2004.05150.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779.

Gordon H. Bower and Daniel G. Morrow. 1990. Mental models in narrative comprehension. *Science*, 247(4938):44–48.

Faeze Brahman and Snigdha Chaturvedi. 2020. Modeling protagonist emotions for emotion-aware storytelling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5277–5294, Online. Association for Computational Linguistics.

Faeze Brahman, Meng Huang, Oyvind Tafjord, Chao Zhao, Mrinmaya Sachan, and Snigdha Chaturvedi. 2021. "let your characters tell their story": A dataset for character-centric narrative understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 1734–1752, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Faeze Brahman, Alexandru Petrusca, and Snigdha Chaturvedi. 2020. Cue me in: Content-inducing approaches to interactive story generation. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 588–597, Suzhou, China. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D 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 Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Paweł Budzianowski and Ivan Vulić. 2019. Hello, it's GPT-2 - how can I help you? Towards the use of pretrained language models for task-oriented dialogue systems. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 15–22.

Orson Scott Card. 1999. *Elements of Fiction Writing - Characters & Viewpoint*. Writer's Digest Books.

Nathanael Chambers. 2013. Event schema induction with a probabilistic entity-driven model. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Seattle, Washington, USA. Association for Computational Linguistics.

Nathanael Chambers and Dan Jurafsky. 2008. Unsupervised learning of narrative event chains. In *Proceedings of ACL-08: HLT*, pages 789–797, Columbus, Ohio. Association for Computational Linguistics.

Nathanael Chambers and Dan Jurafsky. 2009. Unsupervised learning of narrative schemas and their participants. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 602–610, Suntec, Singapore. Association for Computational Linguistics.

Snigdha Chaturvedi, Mohit Iyyer, and Hal Daumé III. 2017a. Unsupervised learning of evolving relationships between literary characters. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence,*, pages 3159–3165.

Snigdha Chaturvedi, Haoruo Peng, and Dan Roth. 2017b. Story comprehension for predicting what happens next. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1603–1614.

Jiaao Chen, Jianshu Chen, and Zhou Yu. 2019. Incorporating structured commonsense knowledge in story completion. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, pages 6244–6251.

Mingda Chen, Sam Wiseman, and Kevin Gimpel. 2021. WikiTableT: A large-scale data-to-text dataset for generating Wikipedia article sections. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 193–209, Online. Association for Computational Linguistics.

Wenhu Chen, Jianshu Chen, Yu Su, Zhiyu Chen, and William Yang Wang. 2020. Logical natural language generation from open-domain tables. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7929–7942, Online. Association for Computational Linguistics.

Liying Cheng, Dekun Wu, Lidong Bing, Yan Zhang, Zhanming Jie, Wei Lu, and Luo Si. 2020. ENT-DESC: Entity description generation by exploring knowledge graph. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1187–1197, Online. Association for Computational Linguistics.

Eric Chu, Prashanth Vijayaraghavan, and Deb Roy. 2018. Learning personas from dialogue with attentive memory networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2638–2646, Brussels, Belgium. Association for Computational Linguistics.

Elizabeth Clark, Yangfeng Ji, and Noah A Smith. 2018a. Neural text generation in stories using entity representations as context. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2250–2260.

Elizabeth Clark, Yangfeng Ji, and Noah A. Smith. 2018b. Neural text generation in stories using entity representations as context. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2250–2260.

Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A. Smith. 2018c. Creative writing with a machine in the loop: Case studies on slogans and stories. In *IUI*, pages 329–340. ACM.

Gregory Curie. 2009. Narrative and the psychology of character. *The journal of aesthetics and art criticism*, 67(1):61–71.

Rajarshi Das, Manzil Zaheer, Siva Reddy, and Andrew McCallum. 2017. Question answering on knowledge bases and text using universal schema and memory networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 358–365. Association for Computational Linguistics.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *8th International Conference on Learning Representations*.

Dorottya Demszky, Kelvin Guu, and Percy Liang. 2018. Transforming question answering datasets into natural language inference datasets. *ArXiv*, abs/1809.02922.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186.

Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. 2019. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895, Florence, Italy. Association for Computational Linguistics.

Esin Durmus, He He, and Mona Diab. 2020. FEQA: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5055–5070, Online. Association for Computational Linguistics.

Susan Dymock. 2007. Comprehension strategy instruction: Teaching narrative text structure awareness. *The reading teacher.*, 61(2).

Paul Ekman. 1992. An argument for basic emotions. *Cognition & emotion*, 6(3-4):169–200.

Micha Elsner. 2012. Character-based kernels for novelistic plot structure. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 634–644, Avignon, France. Association for Computational Linguistics.

David K. Elson, Nicholas Dames, and Kathleen R. McKeown. 2010. Extracting social networks from literary fiction. In *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 138–147. The Association for Computer Linguistics.

Matan Eyal, Tal Baumel, and Michael Elhadad. 2019. Question answering as an automatic evaluation metric for news article summarization. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3938–3948, Minneapolis, Minnesota. Association for Computational Linguistics.

Angela Fan, David Grangier, and Michael Auli. 2018a. Controllable abstractive summarization. In *Proceedings of the 2nd Workshop on Neural Machine Translation and Generation*, pages 45–54.

Angela Fan, Thibaut Lavril, Edouard Grave, Armand Joulin, and Sainbayar Sukhbaatar. 2021. Addressing some limitations of transformers with feedback memory.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018b. Hierarchical neural story generation. *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 889–898.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018c. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.

Angela Fan, Mike Lewis, and Yann Dauphin. 2019. Strategies for structuring story generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2650–2660.

Alvan R Feinstein and Domenic V Cicchetti. 1990. High agreement but low kappa: I. the problems of two paradoxes. *Journal of clinical epidemiology*, 43(6):543–549.

Jeffrey Flanigan, Chris Dyer, Noah A. Smith, and Jaime Carbonell. 2016. Generation from Abstract Meaning Representation using tree transducers. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 731–739, San Diego, California. Association for Computational Linguistics.

Lucie Flekova and Iryna Gurevych. 2015. Personality profiling of fictional characters using sense-level links between lexical resources. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1805–1816.

Maxwell Forbes, Jena D. Hwang, Vered Shwartz, Maarten Sap, and Yejin Choi. 2020. Social chemistry 101: Learning to reason about social and moral norms. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 653–670, Online. Association for Computational Linguistics.

Saadia Gabriel, Chandra Bhagavatula, Vered Shwartz, Ronan Le Bras, Maxwell Forbes, and Yejin Choi. 2021. Paragraph-level commonsense transformers with recurrent memory. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 12857–12865. AAAI Press.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Morton Gernsbacher, Harold Goldsmith, and Rachel Robertson. 1992. Do readers mentally represent characters' emotional states? *Cognition & Emotion*, 6(2):89–111.

Pablo Gervás, Belén Díaz-Agudo, Federico Peinado, and Raquel Hervás. 2005. Story plot generation based on cbr. *In Applications and Innovations in Intelligent Systems XII*, 28(1):33–46.

Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an interactive poetry generation system. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, System Demonstrations*, pages 43–48.

Seraphina Goldfarb-Tarrant, Haining Feng, and Nanyun Peng. 2019. Plan, write, and revise: an interactive system for open-domain story generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Demonstrations*, pages 89–97.

Amit Goyal, Ellen Riloff, and Hal Daumé III. 2010. Automatically producing plot unit representations for narrative text. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 77–86, Cambridge, MA. Association for Computational Linguistics.

Herbert P Grice. 1975. Logic and conversation. In *Speech acts*, pages 41–58. Brill.

Barbara J. Grosz, Aravind K. Joshi, and Scott Weinstein. 1995. Centering: A framework for modeling the local coherence of discourse. *Computational Linguistics*, 21(2):203–225.

Simeng Gu, Wei Wang, Fushun Wang, and Jason H Huang. 2016. Neuromodulator and emotion biomarker for stress induced mental disorders. *Neural plasticity*.

Jian Guan, Fei Huang, Minlie Huang, Zhihao Zhao, and Xiaoyan Zhu. 2020. A knowledge-enhanced pretraining model for commonsense story generation. *Transactions of the Association for Computational Linguistics*, pages 93–108.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. REALM: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*.

Michael Alexander Kirkwood Halliday and Chris tian MIM Matthiessen. 2013. *Halliday's introduc- tion to functional grammar.* Routledge.

P.C. Hogan. 2011. *What Literature Teaches Us about Emotion*. Studies in Emotion and Social Interaction. Cambridge University Press.

Dirk Hovy and Diyi Yang. 2021. The importance of modeling social factors of language: Theory and practice. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 588–602, Online. Association for Computational Linguistics.

Junjie Hu, Yu Cheng, Zhe Gan, Jingjing Liu, Jianfeng Gao, and Graham Neubig. 2020. What makes a good story? designing composite rewards for visual storytelling. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Zhiting Hu, Haoran Shi, Bowen Tan, Wentao Wang, Zichao Yang, Tiancheng Zhao, Junxian He, Lianhui Qin, Di Wang, Xuezhe Ma, Zhengzhong Liu, Xiaodan Liang, Wanrong Zhu, Devendra Singh Sachan, and Eric P. Xing. 2019. Texar: A modularized, versatile, and extensible toolkit for text generation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 159–164.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1587–1596.

Chenyang Huang, Osmar Zaïane, Amine Trabelsi, and Nouha Dziri. 2018. Automatic dialogue generation with expressed emotions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Short Papers)*, pages 49–54.

Mohit Iyyer, Anupam Guha, Snigdha Chaturvedi, Jordan Boyd-Graber, and Hal Daumé III. 2016. Feuding families and former friends: Unsupervised learning for dynamic fictional relationships. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1534–1544, San Diego, California. Association for Computational Linguistics.

Gautier Izacard and Edouard Grave. 2021. Leveraging passage retrieval with generative models for open domain question answering. In *EACL*.

Rachael E Jack, Oliver GB Garrod, and Philippe G Schyns. 2014. Dynamic facial expressions of emotion transmit an evolving hierarchy of signals over time. *Current biology*, 24(2):187–192.

Parag Jain, Priyanka Agrawal, Abhijit Mishra, Mohak Sukhwani, Anirban Laha, and Karthik Sankaranarayanan. 2017. Story generation from sequence of independent short descriptions. *CoRR*, pages 234–242.

Yacine Jernite. 2020. Explain anything like i'm five: A model for open domain long form question answering.

Mandar Joshi, Danqi Chen, Yinhan Liu, Daniel S. Weld, Luke Zettlemoyer, and Omer Levy. 2019a. Spanbert: Improving pre-training by representing and predicting spans. *CoRR*, abs/1907.10529.

Mandar Joshi, Omer Levy, Luke Zettlemoyer, and Daniel Weld. 2019b. BERT for coreference resolution: Baselines and analysis. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5803–5808, Hong Kong, China. Association for Computational Linguistics.

Neel Kant, Raul Puri, Nikolai Yakovenko, and Bryan Catanzaro. 2019. Practical text classification with large pre-trained language models. *CoRR*.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

X. J. Kennedy and D. Gioia. 1983. *Literature: An introduc- tion to fiction. Poetry, Drama, and writing*.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling output length in neural encoder-decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1328–1338.

Evgeny Kim and Roman Klinger. 2019a. Frowning Frodo, wincing Leia, and a seriously great friendship: Learning to classify emotional relationships of fictional characters. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short*

Papers), pages 647–653, Minneapolis, Minnesota. Association for Computational Linguistics.

Evgeny Kim and Roman Klinger. 2019b. Frowning frodo, wincing leia, and a seriously great friendship: Learning to classify emotional relationships of fictional characters. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 647–653. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Catherine Kobus, Josep Crego, and Jean Senellart. 2017. Domain control for neural machine translation. In *Proceedings of the International Conference Recent Advances in Natural Language Processing*, pages 372–378.

Hidetsugu Komeda and Takashi Kusumi. 2006. The effect of a protagonist's emotional shift on situation model construction. *Memory & Cognition*, 34:1548–1556.

Rik Koncel-Kedziorski, Dhanush Bekal, Yi Luan, Mirella Lapata, and Hannaneh Hajishirzi. 2019. Text Generation from Knowledge Graphs with Graph Transformers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2284–2293, Minneapolis, Minnesota. Association for Computational Linguistics.

Vinodh Krishnan and Jacob Eisenstein. 2015a. "you're mr. lebowski, I'm the dude": Inducing address term formality in signed social networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1616–1626, Denver, Colorado. Association for Computational Linguistics.

Vinodh Krishnan and Jacob Eisenstein. 2015b. "you're mr. lebowski, I'm the dude": Inducing address term formality in signed social networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1616–1626, Denver, Colorado. Association for Computational Linguistics.

Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 540–551, Hong Kong, China. Association for Computational Linguistics.

Faisal Ladhak, Bryan Li, Yaser Al-Onaizan, and Kathleen McKeown. 2020. Exploring content selection in summarization of novel chapters. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5043–5054, Online. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics*.

Micheal Lebowitz. 1987. Planning stories. *In Proceedings of the cognitive science society*, pages 234–242.

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213, Austin, Texas. Association for Computational Linguistics.

Yann LeCun, Y Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature*, 521:436–44.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, D. Kontokostas, Pablo N. Mendes, Sebastian Hellmann, M. Morsey, Patrick van Kleef, S. Auer, and C. Bizer. 2015. Dbpedia - a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6:167–195.

VI Levenshtein. 1966. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10(8):707.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020a. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Patrick 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. 2020b. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc.

Boyang Li, Stephen Lee-Urban, George Johnston, and Mark Riedl. 2013. Story generation with crowdsourced plot graphs. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*, page 598–604.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Danyang Liu, Juntao Li, Meng-Hsuan Yu, Ziming Huang, Gongshen Liu, Dongyan Zhao, and Rui Yan. 2020. A character-centric neural model for automated story generation. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018. Generating wikipedia by summarizing long sequences. *CoRR*, abs/1801.10198.

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

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

Fuli Luo, Damai Dai, Pengcheng Yang, Tianyu Liu, Baobao Chang, Zhifang Sui, and Xu Sun. 2019. Learning to control the fine-grained sentiment for story ending generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6020–6026.

Kaixin Ma, Hao Cheng, Xiaodong Liu, Eric Nyberg, and Jianfeng Gao. 2021. Open domain question answering over virtual documents: A unified approach for data and text. *CoRR*, abs/2110.08417.

François Mairesse and Marilyn Walker. 2007. PERSONAGE: Personality generation for dialogue. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 496–503, Prague, Czech Republic. Association for Computational Linguistics.

Mehdi Manshadi, Reid Swanson, and Andrew S. Gordon. 2008. Learning a probabilistic model of event sequences from internet weblog stories. In *Proceedings of the Twenty-First International Florida Artificial Intelligence Research Society Conference, May 15-17, 2008, Coconut Grove, Florida, USA*, pages 159–164. AAAI Press.

Lara J. Martin, Prithviraj Ammanabrolu, , Xinyu Wang, William Hancock, Shruti Singh, Brent Harrison, and Mark O. Riedl. 2018. Event representations for automated story generation with deep neural nets. *In Thirty-Second AAAI Conference on Artificial Intelligence.*

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual*

Meeting of the Association for Computational Linguistics, pages 1906–1919, Online. Association for Computational Linguistics.

Neil McIntyre and Mirella Lapata. 2010. Plot induction and evolutionary search for story generation. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1562–1572, Uppsala, Sweden. Association for Computational Linguistics.

Robert McKee. 2003. Storytelling that moves people: A conversation with screenwriting coach robert mckee. *Harvard business review*, 81:51–5, 136.

Hardik Meisheri and Lipika Dey. 2018. TCS research at SemEval-2018 task 1: Learning robust representations using multi-attention architecture. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 291–299.

Gonzalo Méndez, Pablo Gervás, and Carlos León. 2016. On the use of character affinities for story plot generation. In *Knowledge, Information and Creativity Support Systems*, pages 211–225.

Rada Mihalcea and Hakan Ceylan. 2007. Explorations in automatic book summarization. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 380–389, Prague, Czech Republic. Association for Computational Linguistics.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 404–411, Barcelona, Spain. Association for Computational Linguistics.

Saif Mohammad. 2018. Word affect intensities. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation*, pages 173–183.

Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. 2018. SemEval-2018 task 1: Affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1–17.

Daniel G. Morrow. 1985. Prominent characters and events organize narrative understanding. *Journal of Memory and Language*, 24(3):304–319.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James F. Allen. 2016. A corpus and cloze evaluation for deeper understanding of commonsense stories. *HLT-NAACL*.

Linyong Nan, Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Mutethia Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, Richard Socher, and Nazneen Fatema Rajani. 2021. DART: Open-domain structured data record to text generation. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 432–447, Online. Association for Computational Linguistics.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses: An Introduction*. Wiley New York.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2241–2252, Copenhagen, Denmark. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318.

Ankur Parikh, Xuezhi Wang, Sebastian Gehrmann, Manaal Faruqui, Bhuwan Dhingra, Diyi Yang, and Dipanjan Das. 2020. ToTTo: A controlled table-to-text generation dataset. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1173–1186, Online. Association for Computational Linguistics.

Brian Parkinson and Antony Manstead. 1993. Making sense of emotion in stories and social life. *Cognition and Emotion*, 7(3-4):295–323.

Romain Paulus, Caiming Xiong, and Richard Socher. 2018. A deep reinforced model for abstractive summarization. In *6th International Conference on Learning Representations*.

Nanyun Peng, Marjan Ghazvininejad, Jonathan May, and Kevin Knight. 2018. Towards controllable story generation. In *Proceedings of the First Workshop on Storytelling*, pages 43–49.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Rafael Pérez y Pérez and Mike Sharples. 2001. Mexica: A computer model of a cognitive account of creative writing. *J. Exp. Theor. Artif. Intell.*, 13:119–139.

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics.

Andrew Piper, Richard Jean So, and David Bamman. 2021. Narrative theory for computational narrative understanding. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 298–311, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Robert Plutchik. 1982. A psychoevolutionary theory of emotions. *Social Science Information*, 21(4-5):529–553.

Jullie Porteous and Mike Cavazza. 2009. Controlling narrative generation with planning trajectories: the role of constraints. *In Joint International Conference on Interactive Digital Storytelling*, pages 234–245.

Gerald Prince. 1973. *A Grammar of Stories: An Introduction*.

Ratish Puduppully, Li Dong, and Mirella Lapata. 2019. Data-to-text generation with entity modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2023–2035, Florence, Italy. Association for Computational Linguistics.

Rafael Pérez y Pérez. 2007. Employing emotions to drive plot generation in a computer-based storyteller. *Cognitive Systems Research*, 8:89–109.

Lianhui Qin, Antoine Bosselut, Ari Holtzman, Chandra Bhagavatula, Elizabeth Clark, and Yejin Choi. 2019. Counterfactual story reasoning and generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 5043–5053.

Dragomir R. Radev. 2001. Experiments in single and multidocument summarization using mead. In *In First Document Understanding Conference*.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:8.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020a. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020b. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020c. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. 2010. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 979–988, Uppsala, Sweden. Association for Computational Linguistics.

Steven J. Rennie, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1179–1195.

Leonardo F. R. Ribeiro, Claire Gardent, and Iryna Gurevych. 2019. Enhancing amr-to-text generation with dual graph representations. *CoRR*, abs/1909.00352.

Mark O. Riedl and R. Michael Young. 2010a. Narrative planning: Balancing plot and character. *J. Artif. Int. Res.*, 39(1):217–268.

Mark O. Riedl and Robert Michael Young. 2010b. Narrative planning: Balancing plot and character. *Journal Of Artificial Intelligence Research*.

Melissa Roemmele. 2016. Writing Stories with Help from Recurrent Neural Networks. In *AAAI Conference on Artificial Intelligence; Thirtieth AAAI Conference on Artificial Intelligence*, pages 4311 – 4312, Phoenix, AZ. AAAI Press.

Melissa Roemmele and Andrew S. Gordon. 2015. Creative Help: A Story Writing Assistant. In *Interactive Storytelling*, volume 9445, pages 81–92. Springer International Publishing, Copenhagen, Denmark.

Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. 2010. Automatic keyword extraction from individual documents. *Text Mining: Applications and Theory*, pages 1 – 20.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021a. Efficient content-based sparse attention with routing transformers. *Trans. Assoc. Comput. Linguistics*, 9:53–68.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021b. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

Maarten Sap, Ronan Le Bras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, and Yejin Choi. 2019. ATOMIC: an atlas of machine commonsense for if-then reasoning. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, pages 3027–3035.

Maarten Sap, Saadia Gabriel, Lianhui Qin, Dan Jurafsky, Noah A. Smith, and Yejin Choi. 2020. Social bias frames: Reasoning about social and power implications of language. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5477–5490, Online. Association for Computational Linguistics.

R. Schank and R. Abelson. 1977. *Scripts, plans, goals and understanding: An inquiry into human knowledge structures*. Lawrence Erlbaum Associates, Hillsdale, NJ.

Thomas Scialom, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. 2019. Answers unite! unsupervised metrics for reinforced summarization models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3246–3256, Hong Kong, China. Association for Computational Linguistics.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Zhihong Shao, Minlie Huang, Jiangtao Wen, Wenfei Xu, and Xiaoyan Zhu. 2019. Long and diverse text generation with planning-based hierarchical variational model. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3257–3268.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and J. Weston. 2021. Retrieval augmentation reduces hallucination in conversation. *ArXiv*, abs/2104.07567.

Zhenqiao Song, Xiaoqing Zheng, Lu Liu, Mu Xu, and Xuanjing Huang. 2019. Generating responses with a specific emotion in dialog. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3685–3695.

Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. 2017. Cold fusion: Training seq2seq models together with language models. *arXiv preprint arXiv:1708.06426*.

Shashank Srivastava, Snigdha Chaturvedi, and Tom M. Mitchell. 2016. Inferring interpersonal relations in narrative summaries. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2807–2813.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems*, pages 3104–3112.

Reid Swanson and Andrew S. Gordon. 2012. Say Anything: Using Textual Case-Based Reasoning to Enable Open-Domain Interactive Storytelling. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 2(3).

Pradyumna Tambwekar, Murtaza Dhuliawala, Lara J. Martin, Animesh Mehta, Brent Harrison, and Mark O. Riedl. 2019. Controllable neural story plot generation via reward shaping. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence,*, pages 5982–5988.

Mariët Theune, Sander Rensen, Rieks op den Akker, Dirk Heylen, and Anton Nijholt. 2004. Emotional characters for automatic plot creation. In *Technologies for Interactive Digital Storytelling and Entertainment*, pages 95–100.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: The impact of student initialization on knowledge distillation. *CoRR*, abs/1908.08962.

Hardik Vala, David Jurgens, Andrew Piper, and Derek Ruths. 2015. Mr. bennet, his coachman, and the archbishop walk into a bar but only one of them gets recognized: On the difficulty of detecting characters in literary texts. In *Proceedings of the 2015*

Conference on Empirical Methods in Natural Language Processing, pages 769–774, Lisbon, Portugal. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in Neural Information Processing Systems 30*, pages 5998–6008.

Alex Wang, Kyunghyun Cho, and Mike Lewis. 2020. Asking and answering questions to evaluate the factual consistency of summaries. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5008–5020, Online. Association for Computational Linguistics.

Di Wang, Nebojsa Jojic, Chris Brockett, and Eric Nyberg. 2017. Steering output style and topic in neural response generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2140–2150.

Xin Wang, Wenhu Chen, Yuan-Fang Wang, and William Yang Wang. 2018. No metrics are perfect: Adversarial reward learning for visual storytelling. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 899–909, Melbourne, Australia. Association for Computational Linguistics.

Noah Weber, Leena Shekhar, Heeyoung Kwon, Niranjan Balasubramanian, and Nathanael Chambers. 2020. Generating narrative text in a switching dynamical system. *CoRR*, abs/2004.03762.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256.

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2017. Challenges in data-to-document generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, Copenhagen, Denmark. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019a. Huggingface's transformers: State-of-the-art natural language processing. *arXiv*.

Thomas Wolf, Victor Sanh, Julien Chaumond, and Clement Delangue. 2019b. Transfertransfo: A transfer learning approach for neural network based conversational agents. *CoRR*, abs/1901.08149.

Zeqiu Wu, Michel Galley, Chris Brockett, Yizhe Zhang, Xiang Gao, Chris Quirk, Rik Koncel-Kedziorski, Jianfeng Gao, Hannaneh Hajishirzi, Mari Ostendorf, and Bill

Dolan. 2021. A controllable model of grounded response generation. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 14085–14093. AAAI Press.

Jingjing Xu, Yi Zhang, Qi Zeng, Xuancheng Ren, Xiaoyan Cai, and Xu Sun. 2018. A skeleton-based model for promoting coherence among sentences in narrative story generation. *In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4306–4315.

Kun Xu, Siva Reddy, Yansong Feng, Songfang Huang, and Dongyan Zhao. 2016. Question answering on Freebase via relation extraction and textual evidence. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2326–2336, Berlin, Germany. Association for Computational Linguistics.

Yichong Xu, Chenguang Zhu, Shuohang Wang, Siqi Sun, Hao Cheng, Xiaodong Liu, Jianfeng Gao, Pengcheng He, Michael Zeng, and Xuedong Huang. 2021. Human parity on commonsenseqa: Augmenting self-attention with external attention. *CoRR*, abs/2112.03254.

Lili Yao, Nanyun Peng, Weischedel Ralph, Kevin Knight, Dongyan Zhao, and Rui Yan. 2019. Plan-and-write: Towards better automatic storytelling. *AAAI*.

Meng-Hsuan Yu, Juntao Li, Danyang Liu, Bo Tang, Haisong Zhang, Dongyan Zhao, and Rui Yan. 2020. Draft and edit: Automatic storytelling through multi-pass hierarchical conditional variational autoencoder. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 1741–1748.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020a. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. Personalizing dialogue agents: I have a dog, do you have pets too? In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020b. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Weiwei Zhang, Jackie Chi Kit Cheung, and Joel Oren. 2019. Generating character descriptions for automatic summarization of fiction. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019*, pages 7476–7483. AAAI Press.

Yizhe Zhang, Siqi Sun, Xiang Gao, Yuwei Fang, Chris Brockett, Michel Galley, Jianfeng Gao, and Bill Dolan. 2021. Joint retrieval and generation training for grounded text generation. *arXiv preprint arXiv:2105.06597*.

Hao Zhou, Minlie Huang, Tianyang Zhang, Xiaoyan Zhu, and Bing Liu. 2018. Emotional chatting machine: Emotional conversation generation with internal and external memory. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 730–739.

Xianda Zhou and William Yang Wang. 2018. MojiTalk: Generating emotional responses at scale. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Long Papers)*, pages 1128–1137.