NEURAL GENERATION OF OPEN-ENDED TEXT AND DIALOGUE

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Abigail See
August 2021

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Christopher Manning, Primary Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Tatsunori Hashimoto

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

Dan Jurafsky

Approved for the Stanford University Committee on Graduate Studies.

Stacey F. Bent, Vice Provost for Graduate Education

This signature page was generated electronically upon submission of this dissertation in electronic format. An original signed hard copy of the signature page is on file in University Archives.

Abstract

Advances in Deep Learning have enabled more fluent and flexible Natural Language Generation (NLG). While these neural generative systems achieved early success in machine translation, they encounter problems – such as repetition, incoherence, and uncontrollability – when applied to more open-ended tasks such as abstractive summarization, story generation and chitchat dialogue. Furthermore, open-ended neural generative models tend to be evaluated by crowdworkers in carefully-controlled environments; it is less well-understood how they behave in realistic environments with real-life users. This thesis analyzes and improves neural generative systems performing several open-ended tasks; in the case of dialogue, the systems are evaluated in their full social context.

First, for abstractive summarization, I present a pointer-generator model to improve copying accuracy and a coverage mechanism to reduce repetition in the generated summaries. Next, for chitchat dialogue, I present a large-scale detailed human evaluation to reveal the relationship between bot behaviors (such as repetition, specificity, staying on topic, and question-asking) and human quality judgments, and show that by controlling these bot behaviors, we can improve user experience. Third, for story generation, I characterize the effect of large-scale pretraining, and of the decoding algorithm, on several syntactic, semantic, structural, and stylistic aspects of the generated text. Lastly, I present a study of a neural generative chitchat model in deployment as part of the Alexa Prize, talking to real, intrinsically-motivated users. By analysing bot-user interactions, I identify the bot's main error types, and how they relate to user dissatisfaction. Furthermore, I demonstrate a semi-supervised method to learn from dissatisfaction and thus improve the dialogue system.

Acknowledgments

When I started the PhD program in 2015, I don't think I fully appreciated the magnitude of this six-year undertaking. Undoubtedly, anything that takes six years to complete will have many ups and downs.

I could not achieved these highs, nor weathered these lows, without the wisest and kindest advisor, Chris Manning. When we first met, I didn't know anything about NLP or AI, and I wasn't much of a software engineer either – so the fact that Chris was willing to give me a chance is amazing. As I got to know Chris, I saw that throughout his work, he strives to do the right thing and to do it well. In this way, he has inspired me and supported me to become a better scientist, teacher, community member, and leader. Chris is not only a wonderful advisor, he's also a wonderful person who I enjoy spending time with. His unique perspectives, encyclopedic knowledge of NLP history, and sense of humor brighten every meeting. Especially in the these last two years, there are times when work life is unavoidably affected by events outside work. I am so grateful to Chris for caring, understanding and prioritizing his students' well-being. So thank you Chris, for having the best influence on this period of my life and my career.

Thank you to my reading and oral committee – Dan Jurafsky, Tatsu Hashimoto, Monica Lam and Serena Yeung – whose support, feedback and community have been so valuable throughout my time at Stanford. They are all role models to me, and have made the department a better place.

I've been fortunate to work with many wonderful research collaborators at Google, FAIR, and Stanford. Thang Luong, Peter Liu, Jason Weston, Douwe Kiela, Stephen Roller, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, Kelvin Guu, John Hewitt, and Joyce Xu – these colleagues, mentors and mentees have helped me better understand how to do research.

Going back even further, I also want to thank Byron Cook for welcoming me into research as an undergraduate – I never would have considered a PhD without his encouragement.

Last year I had the opportunity to work on the Alexa Prize with Team Chirpy 1.0: Ashwin Paranjape, Kathleen Kenealy, Haojun Li, Amelia Hardy, Peng Qi, Kaushik Ram Sadagopan, Nguyet Minh Phu, and Dilara Soylu. I am so proud of what this dedicated and passionate team achieved – together we built a chatbot that may not have always made sense, but was definitely interesting. In particular, Ashwin was an amazing co-lead, and it was a great experience to work so closely together. I'm grateful to the Amazon Alexa Prize team for giving us the opportunity to bring conversational AI to real users.

One of the most rewarding things I've done at Stanford is to contribute to our annual class *CS224n: NLP and Deep Learning* – an invigorating experience, as our teaching materials become out-of-date a year after we make them. I loved working with Chris Manning, Richard Socher, Kevin Clark, Sahil Chopra, the TAs, admins, students and SCPD to create and deliver teaching material that brings cutting-edge NLP knowledge to more and more people across the world.

Next, I want to thank the Stanford NLP group. I'm so glad that I got to work in this field during such a transformative period, and there's nowhere better I could have done it than Stanford NLP. It is a lovely community dedicated to mutual learning, friendliness and support – in fact, it is the main reason I chose to work in NLP. I loved sharing office space with the professors, postdocs, admins, students and engineers – I will miss our reading groups, paper swaps, cluster drama, debugging sessions, corridor chats, movie nights, hikes, retreats, and in-jokes.

My research was aided by funding from the NVIDIA, Gerald J. Lieberman, and Google Fellowships, as well as grants from DARPA, Tencent, and Amazon. I also received help from many anonymous people: the reviewers whose feedback improved my work, the Alexa users who interacted with Chirpy, and the crowdworkers and annotators who produced the data we used. This anonymous work powers modern AI.

There are so many fantastic friends I've made over the last few years – Andrey, Drew, Chris, Nathan, Lex, Sharon, Michelle, Ann, Jonathon, Lily, Alyssa, Lucas, Vincent, Ze, Tushar, John, Carolyn, Jeremy, Mandy, Felix, Pujun, Tegan, Aaron, Laurel – these people and more have made my life better in all kinds of ways. In particular, I'm grateful to the

Stanford social dance scene – it is a creative, innovative and inclusive community where I've made friends and tried things I never thought I would. Participating in the Viennese Ball Opening Committee was a once-in-a-lifetime experience. Thank you to everyone who's been there for me and injected these years with fun.

Last, I want to thank my family. First, my sisters Helena and Jessie: they have always been my number one fans who I can rely on to talk about anything, or to cheer me up. I'm so glad that we're now together again. Most of all, I want to thank my parents: Clare, who gave me a love of language, and Thomas, who gave me a love of science. From the first day to the last day of school, they have nurtured my creativity, curiosity and individuality. Even when I decided to go to the other side of the world for six years, they have been nothing but supportive. I have no doubt that my success is due to their parenting.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

For machines to communicate effectively with humans, they must not only understand human language, but generate it too. Indeed, giving voice to the machine can create a powerful effect on users. ELIZA (Weizenbaum, 1966), an early experimental chatbot which used primitive rephrasing and substitution techniques to imitate a psychotherapist, fooled many users into believing it held true intelligence. Of course, ELIZA was an illusion with no real understanding of its conversations – Weizenbaum said that only people who misunderstood ELIZA called it a sensation (Schanze, 2010).

In the 2010s, the Deep Learning renaissance revolutionized Natural Language Processing (NLP). The neural Language Model (LM) – a neural network that takes a piece of text, computes a representation of it, and uses that representation to predict possible next words – became the ubiquitous starting-point for both Natural Language Understanding (NLU) and Natural Language Generation (NLG). The symbiotic relationship between understanding and generation was drawn ever closer: the strength of the neural Language Model lies in its ability to learn representations (i.e., understand) and to predict (i.e., generate) jointly. Neural Language Models provided not only incredible progress on NLU benchmarks (Wang et al., 2018, 2019), but also a method for more flexible, fluent text generation. With increasing model size and training data, neural Language Models are capable of generating longer samples of startlingly convincing text, as well as displaying exciting emergent abilities such as few-shot learning (Brown et al., 2020). These abilities have captured public imagination, sparking debates in both academic and mainstream media (Bender and Koller, 2020; Michael, 2020; Kurenkov and Arora, 2019; Nair and Bashir, 2020) about whether a neural Language Model – even if extremely powerful – could ever constitute Artificial General Intelligence (AGI).

Open-ended text generation encompasses many exciting applications where the output text is less constrained – such as automatic summarization, dialogue, and storytelling. Despite the considerable progress of neural LMs, practical application to open-ended tasks remains tricky. First, though LM-generated text *can* be fluent and convincing, it is not reliably so; nonsensical and incoherent output is common (Tan et al., 2021). In particular, the highest-quality samples are obtained only from the largest LMs, but these systems are inaccessible for many practical applications due to financial and infrastructure limitations (Strubell et al., 2019). Second, though LMs learn to generate a likely sequence of output text given some input, in open-ended settings it is not easy to place constraints on what is generated (for example, the style, structure or topic of the output text). This lack of control also presents safety concerns, as it is difficult to prevent the model from generating offensive or socially biased text (Bender et al., 2021). Third, the quality of open-ended generated text is difficult to quantify – typically requiring human rather than automatic evaluation. Even with human evaluation, there can be a disconnect between the synthetic evaluation settings used in research and the conditions of real-life usage (Celikyilmaz et al., 2020). Together, these problems form a substantial obstacle to the practical deployment of open-ended neural text generation systems.

Now over 50 years since ELIZA, in this thesis I pose the following questions: What are today's open-ended neural generative systems capable of, and how do they interact with modern users? What problems do they face, and how can we tackle them? And lastly, what is required to make them ready for real-life deployment?

In this chapter, I first provide some brief remarks on pre-neural text generation (Section 1.1). Then, starting with the foundational task of Neural Machine Translation (NMT), I identify four main problems in neural text generation (Section 1.2). Moving on to more open-ended generation tasks like dialogue, I explain how these four problems become even more formidable (Section 1.3). Inspired by these problems, I describe the core research questions that I address in this thesis (Section 1.4). Finally, I provide an overview of the thesis (Section 1.5), and summarize its main findings (Section 1.6).

1.1 Pre-neural Text Generation

In their 1998 introduction to the Special Issue on Natural Language Generation, Dale et al. (1998) divide NLG tasks into three typical subparts:

1. **Content determination**: Deciding what to say; typically this involves selecting the right information from what is available.

2. **Text structure**: Determining how the information should be arranged into sentences and paragraphs.

3. **Surface realization**: Mapping the selected content to morphologically and lexically well-formed words and sentences.

Researchers developed various approaches to fulfil these aims in different NLG tasks:

Summarization. In pre-neural summarization (Nenkova and McKeown, 2011), **content determination** mostly amounts to identifying the most important sentences in the source document; this can be achieved via word frequencies, graph-based approaches, or through machine learning using features such as sentence length, cue phrases and document position. **Text structure** is relatively simple for single-document summarization, as the summary sentences can usually be kept in their original order. For multi-document summarization, sentences may be arranged using formalized notions of coherence (e.g., that neighboring sentences should use similar entities). Finally, **surface realization** typically consists of minor edits to the assembled sentences, such as simplification, coreference and other discourse adjustments.

Dialogue. There are many types of dialogue systems including simple rule-based systems like ELIZA, retrieval-based systems, and task-oriented systems (Jurafsky and Martin, 2009). The *dialogue state architecture* is an approach to task-oriented dialogue that was already well-developed in the pre-neural era. In this approach, the task (e.g., restaurant booking) is conceptualized as understanding which of a predefined set of intents the user is trying to achieve (e.g., booking a table, cancelling a booking) and filling the necessary slots with required information (e.g., number of people, time). **Content determination** is handled by a dialogue policy module which decides the next action based on the current state, and **text structure** and **surface realization** are most commonly handled by filling slots in a prewritten template.

Machine Translation. The dominant pre-neural approach to machine translation was Statistical Machine Translation (SMT) (Koehn, 2009). As machine translation was generally regarded as a separate research area to NLG, it did not necessarily follow the same subtask structure described above. SMT involves splitting the translation task into a *translation model* and a *language model*; the translation model is decomposed further to include an *alignment model*. The system learns from data how words and phrases align to other words and phrases, taking into account features like the fertility of individual words and their position in the sentence.

The biggest impact of Deep Learning on NLG research was to **unify the methodology** – potentially replacing these task-specific approaches with a single approach that can be applied to summarization, dialogue, machine translation, and more. The neural approach also simplified the standard NLG pipeline of content determination, text structure, and surface realization; a single neural network learns end-to-end to perform all these subtasks. This unification makes it possible for this thesis to study several different NLG tasks, while focusing on one core technique.

1.2 Neural Text Generation: Early Success in Machine Translation

Neural Machine Translation (NMT) is arguably the most striking example of how Deep Learning enacted a sweeping transformation on NLP, from research to deployment. NMT's rapid ascent was remarkable: in just two years, it went from a research project (Sutskever et al., 2014) to serving millions of users daily[1] in Google Translate (Wu et al., 2016). Compared to the significantly more mature Statistical Machine Translation (SMT) approach (Koehn, 2009), NMT offered a relatively straightforward end-to-end solution that required

[1] `https://ai.googleblog.com/2016/09/a-neural-network-for-machine.html`

significantly less engineering effort than SMT's piecemeal pipeline. Provided with enough training data and computing power, a sufficiently large NMT model outperformed SMT on fluency, use of context and generalization.

Indeed, NMT's success sparked a flurry of work applying the NMT blueprint to other generative tasks, such as summarization (Rush et al., 2015), dialogue (Shang et al., 2015), semantic parsing (Jia and Liang, 2016), syntactic parsing (Vinyals et al., 2015b), source code description (Iyer et al., 2016), and grapheme-to-phoneme conversion (Yao and Zweig, 2015). Framed as *sequence-to-sequence* problems, these tasks involve mapping an input sequence x to an output sequence y (see Section 2.1). Following the NMT blueprint, researchers typically applied Recurrent Neural Network (RNN) sequence-to-sequence models with an attention mechanism (Section 2.3), supervised learning on a parallel corpus, beam search decoding (Section 2.6.1) and evaluation via n-gram overlap based metrics such as BLEU (Papineni et al., 2002) or ROUGE (Lin, 2004b). More recently, Transformers (Vaswani et al., 2017) superseded RNNs – by replacing recurrent connections with attention-based connections throughout, they achieve efficiency improvements and higher performance (see Section 2.3).

Though this thesis does not focus on machine translation as a task, it is important to understand the origins of the NMT blueprint that was later applied to open-ended generation. In this section I describe several tricky generation-related problems in NMT that are subtler than they initially appear. By understanding these problems in the NMT blueprint's original setting, we can understand how they manifest for open-ended generation later (Section 1.3.3).

Problem 1: Choice of decoding algorithm. Though beam search is the standard decoding algorithm in NMT, the details of its implementation have a substantial but sometimes overlooked effect on the quality of the generated text. Britz et al. (2017) report that the choice of beam size and length normalization penalty (see Section 2.6.1) are 'just as crucial as most architectural variations' to achieve good performance. Though beam search is founded on the goal of finding the highest-probability output sequence y given the input x, empirical analysis shows that a larger beam size can lead to worse-quality outputs (Koehn and Knowles, 2017; Cohen and Beck, 2019) – that is, the model can assign higher probabilities to worse-quality outputs. In particular, the model has an inherent bias towards shorter sequences which omit parts of the source text; indeed the globally most likely sequence is often the empty sequence (Stahlberg and Byrne, 2019). These observations have led researchers to reconsider the goal of finding the highest-probability translation. For example, Meister et al. (2020) argue that translations should optimize for *uniform information density* – that is, distributing information equally across the sentence – and that beam search has an inherent bias towards this goal.

Problem 2: Repetition and diversity. When decoding with beam search, NMT systems exhibit several issues that can be described as a lack of diversity. First, the generated outputs sometimes contain repetition within themselves (Mi et al., 2016); this phenomenon is sometimes called *over-translation*. Second, NMT models tend to underestimate the probability of rare words; this can lead to *under-translation* (i.e., leaving out parts of the source sentence) and a tendency towards generic phrasing and vocabulary (Ott et al., 2018). Third, the beam itself (i.e., the set of possible output sequences) tends to contain alternatives that differ only slightly in the last few tokens (Vijayakumar et al., 2018). This is disadvantageous – a more diverse set of valid alternatives can be useful, especially if we require multiple alternatives for a separate reranking or selection process.

Problem 3: Hallucination, bias and uncontrollability. NMT systems are known to hallucinate – that is, produce an output y containing content unfounded on anything in the input x. In some cases, this phenomenon reflects patterns seen in the training data – for example, NMT models sometimes generate pronouns according to stereotypical gender roles, even when the input text does not specify gender (Stafanovičs et al., 2020). Hallucination is also triggered by out-of-domain inputs – for low-resource languages, Google Machine Translate was observed to generate biblical-sounding passages when provided nonsensical input sequences of repeated characters (Gong, 2018). Seemingly, in the absence of a learned transformation to apply for the out-of-domain input, NMT systems default to running the target language model unconditionally – in the case of low-resource languages, this echoes the Bible, an important parallel training text (Vincent, 2019).

Problem 4: Evaluation. BLEU (bilingual evaluation understudy) (Papineni et al., 2002) is the *de facto* standard evaluation metric for machine translation. Based on computing n-gram overlaps between the candidate translation and one or more reference translations, BLEU can capture exact similarities in vocabulary and phrasing, but is less able to recognize valid alternatives with higher-level differences. The metric is sensitive to the number, diversity and style of the reference translations; consequently in some cases BLEU can bias towards rewarding 'translationese' (Freitag et al., 2020). Though BLEU can be useful as a proxy for translation quality, there can be conflicting results: in the WMT19 competition, BLEU and human evaluation do not agree on the top-performing system; in fact the correlation is negative for the top 4 systems (Ma et al., 2019). At least 108 alternative automatic metrics have been proposed for machine translation (Marie et al., 2021) – many attempting to improve on BLEU's brittleness to paraphrasing. However, a survey of many of these metrics reveals that their reliability varies – for example, they are generally more reliable at evaluating high-quality translations (vs low-quality translations), evaluating NMT translations (vs SMT translations), and detecting adequacy errors (vs fluency errors) (Fomicheva and Specia, 2019). No automatic metric has been adopted as widely as BLEU, and the community has increasingly moved towards human evaluation to resolve this problem (Marie et al., 2021).

However, human evaluation is far from a simple gold standard. Läubli et al. (2020) show that evaluation results depend on the choice of annotators (for example, crowdworkers or professionals), their access to the document context, and the quality and style of the reference translations under comparison. Though there are many differing ways to conduct human evaluation of machine translation systems, establishing a standard methodology remains a matter of debate (Freitag et al., 2021). In addition, due to the relative inconvenience and cost of human evaluation, the majority of NMT papers continue to (over)rely on BLEU; sometimes drawing conclusions based on BLEU comparisons without adequate significance testing or controlling important hyperparameters (Marie et al., 2021).

Consequently, progress is NMT is hard to measure. Though it's clear that the state-of-the-art continues to make progress overall, finegrained comparisons remain opaque.

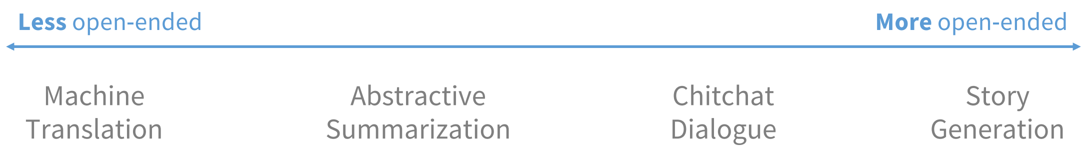


Figure 1.1: Natural Language Generation (NLG) tasks exist on a spectrum from less open-ended to more open-ended.

1.3 Open-ended Neural Text Generation

Having laid the groundwork of NMT, we now move on to open-ended neural text generation. In this section I define open-ended text generation, and summarize the main developments of the field since the early NMT blueprint described in the previous section. Lastly, I detail how the four main generation problems in NMT become even more challenging in the open-ended setting.

1.3.1 What is Open-ended Text Generation?

In the years following its breakout success in NMT, neural NLG research has expanded to many other application areas. As shown in Figure 1.1, some of these tasks are more *open-ended* – meaning there is a greater variety of valid outputs y for a single input x.

As any human translator will attest, translation involves significant nuance and choice on the part of the translator (Taylor, 1998). Nonetheless, abstractive summarization – which involves writing an original summary of a (possibly lengthy) input text – requires even more choice, as the summarizer chooses not only which information to include, but how to phrase it. Story generation – the task of writing or completing a narrative – is perhaps one of the most creative and open-ended text generation tasks.

Chitchat dialogue, which can be defined as an informal but polite social conversation, is a particular focus of this thesis. In contrast to task-oriented dialogue, in which two speakers work together to achieve a goal (such as booking a restaurant, or negotiating a sale), chitchat's goal is purely social. Though certain topics are customary in particular cultures,[2] and one's conversational partner chooses some topics, a conversational agent has significant freedom in determining the direction of the conversation. Chitchat dialogue has many variants including symmetric and asymmetric roles, public or private settings, two or more participants, synchronous or asynchronous communication, introductory chat with a new acquaintance or catching up with an existing friend.

1.3.2 Development

Neural generation for open-ended tasks began with the NMT blueprint described in Section 1.2, but the standard methodology has developed in recent years.

Open-ended neural text generation's greatest divergence from the NMT blueprint is the choice of decoding algorithm. Though beam search performs well for machine translation, researchers found that it tends to produce repetitive and generic text for more open-ended tasks (Holtzman et al., 2020). For these tasks, researchers began to move towards sampling-based decoding algorithms such as top-k sampling (Fan et al., 2018b) and top-p (aka nucleus) sampling (Holtzman et al., 2020) – see Section 2.6.2. Unlike beam search, which searches for the single highest probability output sequence, these randomized algorithms sample each successive token from a truncated distribution containing multiple high-probability alternatives. These algorithms can produce a diverse set of alternative outputs given one input, and generally produce more interesting and varied text.

Following the comprehensive success of ELMo (Peters et al., 2018) and BERT (Devlin et al., 2019) on NLU tasks, pretrained Language Models – trained on large amounts of unlabelled text from diverse web sources – became the indispensable starting-point for finetuning on downstream tasks. Pretrained Language Models can generally be divided into fill-in-the-gap *masked LMs*, which predict one or more masked-out tokens, and *decoder LMs*, which predict tokens left-to-right, and can thus be used to generate text – see Section 2.2. Of the pretrained generative LMs, the GPT series (Radford et al., 2018, 2019; Brown et al., 2020) is the most influential. By scaling up the model size and the quantity of training data

[2]The customary topics may still be debated. In George Bernard Shaw's *Pygmalion*, Higgins instructs Eliza Doolittle – the ELIZA chatbot's namesake – to 'keep to two subjects: the weather and everybody's health. [...] That will be safe.' Mrs. Higgins: 'Safe! To talk about our health! about our insides! perhaps about our outsides! How could you be so silly, Henry?'

(drawn from a broad variety of web sources), the GPT models have demonstrated striking emergent abilities such as closed-book question answering and few-shot learning of several tasks (Brown et al., 2020). Equally importantly, when given a prompt (i.e., some starting text to complete), they are able to generate (via top-k or top-p decoding) samples of text exhibiting previously-unseen coherence over long sequences. Owing to the variety of the training data, the models can match the genre and style of many input prompts, writing convincing news articles, recipes, and blog posts (Radford et al., 2019). For the largest GPT-3 model, human annotators' ability to distinguish whether a 500-word news article is model-generated or human-written is barely above chance (52%) (Brown et al., 2020).

Open-ended neural text generation has taken some tentative steps from research to application. Google Smart Compose (Chen et al., 2019) uses a neural LM to generate real-time auto-complete suggestions for email – for example the prefix *Thanks for the invitation, I'm* could be completed with *afraid I can't make it* or *looking forward to it*. Though this does constitute somewhat open-ended generation, the constraints of the application – requiring only occasional suggestions of short, common and highly likely phrases – limit the open-endedness, thus providing a good fit for established neural generation abilities. More recently, OpenAI released an API[3] providing commercial access to GPT-3's generation abilities, though not the parameters of the model itself. Developers are using the API to build apps for a variety of use-cases, including customer support, semantic search, copywriting, creative writing and games.[4] OpenAI also developed Codex, a GPT-3 model finetuned on publicly available code (Chen et al., 2021) – this powers Github Copilot, an assistive code development tool currently in technical preview.[5] These examples are not yet the norm however, and at the time of writing, open-ended neural text generation is not widely used in deployment beyond NMT.

1.3.3 Difficulties

I argue that progress on practical open-ended neural text generation is impeded by intensified versions of the NMT problems described in Section 1.2. Together, these inter-related issues

[3] https://openai.com/blog/openai-api/
[4] See https://openai.com/blog/gpt-3-apps/ and https://gpt3demo.com/
[5] https://copilot.github.com/

make open-ended generation one of the most uncharted and unmastered areas of Deep Learning NLP.

Problem 1: Choice of decoding algorithm. Although open-ended neural text generation has largely moved towards sampling-based decoding (as described in Section 1.3.2), the choice of decoding algorithm remains a topic of unresolved debate. Some researchers argue that, especially for more creative tasks, the highest-probability sequence is neither human-like nor the right objective, and bound to result in degenerate text (Holtzman et al., 2020). Others argue that, especially for more factual settings like abstractive summarization or question-answering, generation requires precision – the next word of *The capital of Japan is* should be the LM's top-1 most likely option (hopefully *Tokyo*), not sampled from several options (Keskar et al., 2019).

The introduction of top-k and top-p decoding has introduced an even greater choice of decoding algorithm for open-ended generation. The parameter k or p itself has a huge effect, inducing a tradeoff from repetition and genericness (Problem 2) to nonsense and hallucination (Problem 3) (Massarelli et al., 2020). The temperature parameter, which alters the next-token probability distribution to be more or less 'spread out', has a similar effect (Section 2.6.3). Ideally, we should evaluate generative systems across this entire tradeoff curve, not just points on it (Caccia et al., 2018; Hashimoto et al., 2019) – however, doing so can be prohibitively expensive, particularly as identifying nonsense and hallucination tends to require human evaluation. Consequently, we don't have a good understanding of whether top-k, top-p or temperature provides the best quality-diversity tradeoff. Overall, the decoding algorithm is an extremely influential component of open-ended text generation, but also a complicating factor that increases the difficulty of evaluation (Problem 4).

Problem 2: Repetition and diversity. When researchers first applied the NMT blueprint (Section 1.2) to open-ended tasks like dialogue, repetition and genericness were immediately identified as prominent problems (Jiang and de Rijke, 2018). At first, most solutions focused on fixing this issue through the LM itself – e.g., changing the objective function to Maximum Mutual Information (Li et al., 2016a), explicitly modeling topic similarity (Baheti et al., 2018), or providing relevant content words as additional input to the model (Xing et al., 2017). However, researchers soon realized that likelihood-maximizing decoding algorithms like greedy and beam search are a primary cause of this behavior (Holtzman et al., 2020). Sampling-based decoding thus provided a more systematic solution to the repetitiveness and genericness problem – however, it introduces the quality-diversity tradeoff. Meanwhile, those who champion likelihood-maximizing decoding as the principled choice have devised other methods such as unlikelihood training (Welleck et al., 2020) and repetition-penalized sampling (Keskar et al., 2019) to remove repetition and genericness from the most likely sequence.

Problem 3: Hallucination, bias and uncontrollability. In tasks like machine translation, the input x exerts strong constraints on y; thus x functions as a type of control. In more open-ended tasks, x provides less of a framework for y – consequently neural models often generate incoherent, nonsensical, or self-contradictory output, particularly when generating long text or dialogues (Welleck et al., 2019; Tan et al., 2021). Furthermore, when the model has more freedom to generate y, there is greater scope for hallucinated (Maynez et al., 2020; Shuster et al., 2021), off-topic (Xu et al., 2018; Sankar et al., 2019; Jaques et al., 2019), biased (Bordia and Bowman, 2019; Liu et al., 2020; Dinan et al., 2020), and offensive (Wallace et al., 2019; Peng et al., 2020; Gehman et al., 2020) output. While some of these behaviors can be reduced by a conservative (i.e., low) choice of p or k in top-p or top-k sampling, this solution is unsatisfactory as it pushes the model towards genericness.

Pretrained LMs, which tend to be trained on extensive and varied internet datasets, introduce additional concerns. First, powerful pretrained LMs can be manipulated to regurgitate some passages of training data, raising security and privacy risks (Carlini et al., 2020). Second, by casting a wide net in their training data, these LMs learn a pervasive range of undesirable biases and toxicity, which are detectable in their predictions (Huang et al., 2020; Brown et al., 2020; Nadeem et al., 2020), generated text (Sheng et al., 2019) and dialogue (Sheng et al., 2020). Though bias in LMs is a flourishing research area, the issue is very far from simple. There are many types and alternative definitions of bias (Crawford, 2017; Abbasi et al., 2019), often requiring debatable assumptions of what is socially and technologically desirable (Green, 2019). Though the composition of the training data is indeed a major source of bias, other parts of the NLP pipeline – such as task formulation, choice of annotators, annotation instructions, learning objectives, and evaluation metrics – are important factors (Shah et al., 2020). Researchers have proposed many different methods and benchmarks to measure and to mitigate LMs' bias towards different social groups – see Garrido-Muñoz et al. (2021) or Sun et al. (2019) for a survey. However, these efforts have been criticized for a technology-centric view which under-emphasizes the real-life context and actual harms of biased systems on the lived experiences of marginalized groups, as well as the wider philosophical and social science literature (Prabhumoye et al., 2019; Blodgett et al., 2020; Boyarskaya et al., 2020).

Problem 4: Evaluation. At first following the NMT blueprint, researchers evaluated open-ended generative systems with n-gram overlap based metrics such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004b), and METEOR (Denkowski and Lavie, 2014). However, the community soon found that these metrics are deeply flawed for abstractive summarization (Kryściński et al., 2019) and close to useless for very open-ended tasks like dialogue (Liu et al., 2016; Novikova et al., 2017). Though there have been many attempts to create new automatic metrics – in particular, trained model-based metrics such as ADEM (Lowe et al., 2017) and BLEURT (Sellam et al., 2020) – the only consensus to emerge is the necessity of human evaluation. However, human evaluation is also very complex – results can be affected by the choice of evaluation criteria and the rating scale used to measure it; the number, training and demographics of the evaluators; and the order in which they answer the questions (Van Der Lee et al., 2019). As a result, researchers currently use a variety of human evaluation methodologies for NLG, with active debate to determine best practices (Van Der Lee et al., 2019; Li et al., 2019; Clark et al., 2021). Together, these evaluation problems seriously hinder the clarity of open-ended NLG research. For more detail on all these current issues, refer to the survey by Celikyilmaz et al. (2020).

A note on scale. One prominent variable, which affects all four of these problems, is scale – of both the LM and its training data. In this thesis we study these problems with moderate-scale LMs, as most people are not able to access extremely large LMs due to financial and infrastructure constraints (Strubell et al., 2019). However, empirical evidence strongly indicates that scale provides a better quality-diversity tradeoff – that is, a more powerful LM can write more unusual text without risking nonsense (Adiwardana et al., 2020; Brown et al., 2020). Scale also provides much improved coherence for long text (Problem 3), though it is debatable to what extent that coherence reflects true understanding or communicative intent (Bender et al., 2021). It remains to be seen which problems can and cannot be solved with scale; however I provide my opinion in the last part of Section 1.6.

1.4 Research Questions

Motivated by the four problems in the previous section, this thesis focuses on addressing the following three Research Questions:

RQ1: What problems occur in open-ended neural text generation, and how do they affect user experience? Though it can be convenient to distill a system's performance into a single overall evaluation metric, over-relying on such metrics can be misleading, sometimes giving an inflated impression of a system's performance. Whether automatic or human, these one-dimensional evaluation metrics often fail to capture the complex inter-related problems and tradeoffs described in the previous section.

Throughout this thesis, I strive to deeply understand all major strengths and weaknesses of each system, and if possible identify the causes of these issues. I argue that providing a candid, holistic view is an important and valuable contribution to the research community: first, to set realistic expectations for those who may wish to use the system, and second, to highlight important areas for improvement, that may be under-emphasized by simple evaluation metrics alone.

For highly interactive applications like dialogue – in which the user comprises half the system – it is indispensable to understand the interplay between the user's and the chatbot's behaviors. Looking beyond simple binary evaluations of out-of-context utterances, I study the varied and contradictory ways that real people interact with chatbots over whole conversations. Through this integrated perspective, I draw connections between technical neural generative problems and overall user experience.

RQ2: What techniques can we use to address these problems? To tackle the problems outlined in Section 1.3.3, this thesis proposes and evaluates techniques for neural generation across several application areas.

In some of my work, I focus on understanding the effect of standard techniques. In Chapter 5, I investigate the effects of large-scale pretraining and of the decoding algorithm (Problem 1) on aspects of storytelling, including diversity (Problem 2) and coherence (Problem 3).

In other work, I propose extensions to standard techniques. Chapters 3 and 4 both describe modifications of the RNN-based NMT blueprint (Section 1.2). In abstractive summarization (Chapter 3), I propose methods to copy words from the input (addressing Problem 3, hallucination) and to avoid summarizing the same input passages repeatedly (addressing Problem 2, repetition). In chitchat dialogue (Chapter 4), I apply two methods to control low-level attributes of the generated text, including repetition and genericness (Problem 2), staying on topic (Problem 3) and question-asking.

Lastly in Chapter 6, I focus on tackling these problems in real-life deployment. Moving away from techniques targeted to specific problems, I propose a holistic solution that learns to avoid many types of poor-quality bot utterances that lead to user dissatisfaction. This semi-supervised method learns from real user interactions, and thus could be used as an online learning method to continuously improve a bot in deployment.

RQ3: What is required for neural generative dialogue models to be ready for real-life deployment? Recent publications (Zhang et al., 2020b; Adiwardana et al., 2020) show that ever-larger neural generative dialogue models are trending closer to some notion of human performance.[6] These results are typically obtained through crowdworkers having written conversations with the bot in carefully-controlled environments. The crowdworker usually operates from a set of instructions that may define the topic, style, aim, and length of the conversation, and may even include a persona and scenario to role-play (Zhang et al., 2018b). The crowdworker's incentive is external; to receive payment they are incentivized

[6] In chitchat dialogue, 'surpassing human performance' can mean passing a Turing test (i.e., being indistinguishable from a human conversationalist), or surpassing a human conversationalist in some quality evaluation. Both tests are highly dependent on the human(s) conversing with and evaluating the chatbot, as well as the topic, length and other constraints of the conversation.

to carry out these instructions regardless of their personal opinions or experience.

By contrast, real-life deployment scenarios such as the Alexa Prize (Gabriel et al., 2020) present different challenges. Unlike crowdworkers, users have their own expectations and personalities, which vary greatly between users. Their incentive is internal; they take part in the conversation to satisfy their own personal goal of entertainment, curiosity, social interaction or otherwise. Consequently, user behavior can be more varied, unpredictable, and uncooperative than crowdworkers'. Secondly, real-life conversations may be written or spoken. Particularly for spoken conversations, background noise can be a substantial problem – ASR errors, cross-talk and user distraction can obscure the user's conversational intent. Thirdly, latency is more critical in real-life deployment, as real users are less likely to wait patiently for a bot response, and prompt back-and-forth is necessary to maintain spoken conversational flow.

It is less well-understood how neural generative dialogue models perform under these conditions – yet this knowledge is necessary if such models are ever to achieve widespread practical use. In Chapter 6 we deploy a neural generative model in the Alexa Prize. This setting allows us to evaluate neural generative models under realistic cost and latency constraints, identify the generative errors most detrimental to user experience, and understand the environmental and user behaviors that trigger them.

1.5 Thesis Overview

The rest of the thesis is structured as follows. Chapter 2 provides background information on text generation tasks, neural Language Models and decoding algorithms.

Chapter 3: Summarization with Pointer-Generator Networks. Unlike *extractive* text summarization, which involves simply selecting and rearranging passages of the input text to form a summary, *abstractive* text summarization involves writing original text. This chapter proposes two augmentations to the RNN-based sequence-to-sequence attentional model with beam search, which at the time of this work, had recently been applied to abstractive text summarization for the first time. The standard attentional model has two prominent shortcomings: it is liable to reproduce factual details inaccurately, and it tends to repeat itself.

To address the first problem, we use a hybrid pointer-generator network that can copy words from the source text via *pointing*, which aids accurate reproduction of information, while retaining the ability to produce novel words through the *generator*. For the second problem, we use *coverage* to keep track of what has been summarized, which discourages repetition. On the CNN / Daily Mail summarization dataset, our model achieved ROUGE scores that surpassed the abstractive state-of-the-art at the time. This work was first published as See et al. (2017).

Chapter 4: Controlling Attributes of Chitchat Dialogue. A good conversation requires balance – between simplicity and detail; staying on topic and changing it; asking questions and answering them. When applied to chitchat dialogue, RNN sequence-to-sequence attentional models with beam search don't always balance these factors well – for example, generating bland responses, straying off-topic, or asking too few questions. Although dialogue agents are commonly evaluated via human judgments of overall quality, these overall judgments fail to capture different aspects of conversational quality, and how they relate to particular bot conversational behaviors. In this chapter, we apply two existing methods for controlling the output of neural generative models: conditional training and weighted decoding. We use these methods to control four important attributes for chitchat dialogue: repetition, specificity, response-relatedness and question-asking. We conduct a large-scale human evaluation to measure the effect of these control parameters on multi-turn interactive conversations on the PersonaChat task. We provide a detailed analysis of their relationship to high-level aspects of conversational quality, and show that by controlling combinations of these variables our models obtain clear improvements in human quality judgments. This work was first published as See et al. (2019b).

Chapter 5: The Effect of Pretraining for Story Generation. In this chapter we shift focus to Transformer-based language models pretrained on massive amounts of text, which have emerged as a formidable strategy for Natural Language Understanding tasks. However, at the time of this work, their strength as Natural Language *Generators* was less well-understood. Though anecdotal evidence suggested they were capable of generating better quality open-ended text with sampling-based decoding, there had been no detailed study characterizing their generation abilities. In this chapter, we compare an extensively pretrained model, GPT2-117 (Radford et al., 2019), to a similarly-sized but not extensively pretrained state-of-the-art neural story generation model (Fan et al., 2018b). We evaluate the generated text across the top-k decoding spectrum, from greedy decoding ($k = 1$) to natural sampling ($k =$ vocabulary size). Using a wide variety of automatic metrics, we evaluate many syntactic, semantic, structural and stylistic aspects of narrative text generation. We find that GPT2-117 conditions more strongly on context, is more sensitive to ordering of events, and uses more unusual words. However, it is just as likely to produce repetitive and under-diverse text when using likelihood-maximizing decoding algorithms. This work was first published as See et al. (2019a).

Chapter 6: User Dissatisfaction in Chitchat Dialogue. In this chapter we focus on applying open-ended neural generative models to real-life dialogue. In recent years, large Transformer-based generative dialogue agents have shown an increasing ability to hold short chitchat conversations, when evaluated by crowdworkers in controlled settings. However, their performance in real-life deployment – talking to intrinsically-motivated users in noisy environments – is less well-explored. In this chapter, we perform a detailed case-study of a GPT2-based dialogue model deployed as part of Chirpy Cardinal, an Alexa Prize socialbot. In this noisy environment, we find that unclear user utterances are a major source of generative errors such as ignoring, hallucination, unclearness and repetition. However, even in unambiguous contexts the model frequently makes reasoning errors. We study the relationship between these errors and user expressions of dissatisfaction. Though users express dissatisfaction in correlation with these bot errors, certain dissatisfaction types (such as offensiveness and privacy objections) seem to depend on additional factors – such as the user's personal attitudes, and prior unaddressed dissatisfaction in the conversation. Finally, we show that dissatisfied user utterances can be used as a semi-supervised learning signal to improve the dialogue system. We train a model to predict the likelihood that a bot utterance in context will result in next-turn user dissatisfaction. Through human evaluation we find that as a ranking function, the predictor model can be used to select higher-quality neural-generated utterances. This work was first published as See and Manning (2021), and was nominated for a Best Paper Award at SIGDIAL 2021.

Finally, Chapter 7 provides some final thoughts and future directions.

1.6 Main Findings

Though Chapters 3–6 each contain their own conclusions, in this section I provide the main takeaways that I wish the reader to gain from this thesis as a whole.

Taking a wider view. As explained in RQ1 (Section 1.4), the Machine Learning community has a tendency to frame progress in terms of state-of-the-art performance on standardized benchmarks. This thesis demonstrates that a wider view is necessary to understand open-ended text generation systems, which have rich and detailed output. In Chapter 5, we measure several aspects of generated text across the decoding algorithm spectrum. Our analysis emphasizes that Problem 2 (repetition and diversity; see Section 1.3.3) is primarily caused by the decoding algorithm rather than the LM itself, as was previously proposed (Jiang and de Rijke, 2018). In Chapter 4, we go beyond the standard single-measure human evaluation for the ConvAI2 benchmark challenge; in doing so we uncover new complexities – for example, that while our models approach human-level engagingness, they are far from convincingly human. Further scrutinizing this result reveals that 'human-level engagingness' as defined by crowdworkers is problematic, due to the inherently unnatural and extrinsically motivated evaluation task.

The Machine Learning research community also has a tendency to value work on model development more highly than everything else (Sambasivan et al., 2021). However, I have found that 'everything else' – which includes practical considerations like data, hyperparameter tuning, efficiency, user experience and real-life context – is indispensable, both to build something that actually works, and to understand it well. In Chapter 6, these practical considerations bring priorities for neural generative dialogue into sharper focus. We find that larger LMs' latency is prohibitively high for deployment. We also find that the noisy spoken setting and the unpredictable behavior of intrinsically-motivated users creates challenging test-time conversations very different to the training data. This domain shift problem highlights the importance of creating more publicly-available dialogue datasets, improving transfer learning and robustness, and developing online learning.

Dialogue systems in social context. Fundamentally, a dialogue system exists to talk to people, and so must be understood in its social context (see RQ1, Section 1.4). By focusing on this context, in this thesis I discover social problems in neural generative dialogue systems. For example in Chapter 4, we uncover the importance of social behaviors like asking the right amount of questions, and how often to change the topic. In our Alexa Prize work (Chapter 6) the social context is much weightier: our chatbot is in the homes of real people, discussing their lives during an unprecedented pandemic – these conversations are both literally and metaphorically 'kitchen table conversations'. We find that social and contextual awareness are extremely important in this setting, and that our current neural generative methods (in particular their reasoning abilities), are often too rudimentary to grasp this complicated context. In addition we find that users seem to have varying attitudes and motivations for engaging in topics deemed offensive or private.

Understanding social context can provide solutions as well as problems. In Chapter 6, we find that by targeting particular user behaviors, relatively simple adjustments can raise the overall performance of the neural generative system. For example, we find that starting conversations with easy-to-answer open-ended questions, sometimes preceded by self-disclosure (Paranjape et al., 2020), helps users more effortlessly give longer responses; this in turn improves the neural model's performance. In addition, we find that tailoring these starter questions to match the topics of the training data, and providing users clear direction with questions in general, helps keep users on-topic in the model's area of expertise.

Progress, scale, and open science. If we wish to find an accurate and nuanced wider view of the field's progress, it must be done collaboratively – and open science and reproducibility are our best tools to achieve it. Releasing models, code and generated text allows other researchers to analyze NLG systems in new ways that may not have been apparent or possible for the original researchers. Indeed, the released materials for the pointer-generator model (Chapter 3) were used in studies which improved, compared, and highlighted problems with the model, its output, and evaluation (Kryściński et al., 2019; Fabbri et al., 2021). Though releasing models and code is common practice in NLP research, releasing the full generated text of NLG systems (i.e., all the output that was used to produce the analyses and results in the paper) is not. I strongly argue that releasing generated text should become common practice: compared to releasing models and code, which often requires a substantial amount of time and effort, releasing generated text requires almost none. Whether or not the models and code are released, releasing the generated text provides readers an instantly-accessible and infinitely better sense of how an NLG system functions, compared to reading a paper alone (which due to space constraints typically contains only a handful of examples, often cherry-picked). There is however a tension between reproducibility, and the real-user studies advocated in this thesis. Though the Alexa Prize provided a unique opportunity to build a chatbot to interact with a great volume of real users, we are unable to release the data described in Chapter 6.

The advent of large-scale pretrained LMs has affected progress and open science, for better and worse. Pretrained LMs undoubtedly constitute significant progress in many areas; they provide a more powerful starting-point for many researchers and practitioners, and toolkits like HuggingFace Transformers (Wolf et al., 2020) make the process much easier. However, many of the findings and techniques of the preceding years (such as Chapters 3 and 4, and the alternative training objectives in Section 2.5) have been subsumed by increasingly large pretrained Transformers trained with maximum likelihood estimation; this homogenization can in fact limit discovery (Bommasani et al., 2021). Given the widespread influence of large pretrained LMs, it is important that we understand their behaviors, strengths and weaknesses. However, scrutiny of the largest LMs is inaccessible to most researchers outside of well-funded industrial labs, due to financial and compute constraints, as well as unreleased models and training data. Indeed, the ability to analyze and use the leading pretrained LMs has been out-of-reach throughout the work presented in this thesis – in Chapters 5 and 6, our analyses are limited to the smaller GPT-2 models.

Looking to the future, APIs to large pretrained LMs such as OpenAI's may offer increased (paid) access to a certain type of experimentation and usage. Efforts to achieve large LMs' performance in a smaller computational footprint (Menghani, 2021) will both increase access and mitigate their environmental impact (Strubell et al., 2019; Bender et al., 2021). Lastly, academic initiatives such as the National Research Cloud[7] and the Center for Research on Foundation Models (Bommasani et al., 2021) may pave the way to academia playing a greater role in the development, understanding and usage of such models.

[7] https://hai.stanford.edu/policy/national-research-cloud

Control, safety and real-life deployment. Chapters 4 and 6 demonstrate that our neural generative dialogue models are not yet a reliably functional user experience. Indeed, all four of the problems outlined in Section 1.3.3 remain significant impediments to real-life deployment. However, I argue that Problem 3 (control and safety) is the most critical problem to solve for open-ended neural generation to become practically usable.

Control is a theme throughout this thesis. Chapters 3 and 4 both attempt to achieve types of control via targeted modeling interventions – though these can change the targeted behavior, they are tricky to calibrate appropriately. Safety can be framed as a type of control – for example, controlling the model to produce factually correct, non-offensive and unbiased text. While factual faithfulness is an active research area with promising progress (see Section 3.9), in my opinion bias and offensiveness is comparatively much harder to resolve, both philosophically and technically (see Section 1.3.3). Nevertheless, this aspect of safety is crucial for wider deployment of chitchat systems. While Chirpy Cardinal uses crude methods (such as offensive word lists) to refuse to talk about sensitive and controversial issues (Paranjape et al., 2020), such avoidance prevents legitimate discussion of important topics and may marginalize users who wish to discuss them.

Arguably, Problems 1, 2 and 4 (Section 1.3.3) could be sufficiently resolved by increased scale, to allow real-life deployment. But scaling up the current recipe for pretrained LMs will not, in my opinion, resolve the safety and control issue. First, as explained in Section 1.3.3, large-scale pretraining can introduce *new* safety concerns. Second, even if we obtained a 'perfect' LM – that is, one whose conditional P_{LM} distribution exactly matches the output distribution of a well-trained human – we would still require control in most practical applications. For example, we may wish to define policies, rules and constraints on a dialogue agent's topics and conversational structure. For this reason, I believe that control is the central current challenge in open-ended neural text generation.

Chapter 2

Background

In this chapter, I provide some technical background on text generation tasks, neural LMs and how they're trained, and decoding algorithms used to generate text. This provides key material for the methods used in the following chapters.

2.1 Defining Text Generation Tasks

Most text generation tasks can be framed as the task of generating output text y given some input x. The input x may be text (e.g., machine translation), but it may be other modalities too – for example, in image captioning, x is an image. The input x may be a tuple of several, possibly multi-modal inputs – for example, a dialogue agent that discusses an image may take both the image and the conversational history as input.

Some tasks involve generating a continuation of some provided text – for example, story generation in Chapter 5. Formally, this can be expressed as an input prompt $y_1, \ldots, y_t$, and the task of generating a continuation $y_{t+1}, \ldots, y_T$.

2.2 Types of Neural Language Model

Generally, a *language model* is a system which assigns a probability to a text sequence; a *neural language model* uses a neural network to achieve this. Neural language models fall into several main types:

Single decoder LM. These language models assign probability $P_{LM}(y)$ to a text sequence y. Typically, the probability is factorized using the left-to-right chain rule:

$$P_{LM}(y_1, \ldots, y_t) = \prod_{i=1}^{t} P_{LM}(y_i|y_1, \ldots, y_{i-1}) \tag{2.1}$$

These language models can be used to generate text (or continuations of text), or to assign a probability to a sequence of text (e.g., for ranking). They can also be used for their *representations* – that is, the embeddings of the deep neural network form a representation of the input text y. This can be powerful starting-point to provide as input to other deep networks performing other tasks on y.

Encoder-decoder models. These language models assign probability $P_{LM}(y|x)$ to a text sequence y given some input x (which may be text or other modalities, as discussed earlier). There are two parts: an *encoder*, a neural network which takes the input x and produces some representation of it, and a *decoder*, a neural LM which uses those representations to produce $P_{LM}(y|x)$. As with single decoder LMs, the probability is typically factorized left-to-right:

$$P_{LM}(y_1, \ldots, y_t|x) = \prod_{i=1}^{t} P_{LM}(y_i|y_1, \ldots, y_{i-1}, x) \tag{2.2}$$

These models can be used to generate output text y conditioned on some input x, or to compute the conditional probability of a given (x, y) pair.

Masked LM. Masked language models are 'fill-in-the-blank' models that predict the identity of a masked token in a text sequence y. For example, given the input *the children played* `<MASK>` *in the field*, what are the most likely tokens to replace `<MASK>`? Formally, given a text sequence $y_1, \ldots, y_{t-1}, y_{t+1}, \ldots, y_T$, masked language models predict a probability:

$$P_{MLM}(y_t|y_1, \ldots, y_{t-1}, y_{t+1}, \ldots, y_T) \tag{2.3}$$

While decoder LMs predict the next token given the left context, MLMs predict using *bidirectional* context; this is important for learning a strong representation of the input text

y (Devlin et al., 2019). In fact, MLMs are currently the dominant approach to pretraining powerful representations of text in applications where a generative model is not required (e.g., most NLU applications).

As MLMs do not straightforwardly assign a probability to the whole sequence y, they do not strictly fit the definition of Language Model provided above; nonetheless, they are widely referred to as a type of Language Model. MLMs are not generally intended or used to generate text, so they are not a focus of this thesis. However, there are some exceptions – by formulating a MLM as Markov Random Field LM, Wang and Cho (2019) demonstrate a method to generate text from it.

2.3 Architectures for Neural Language Models

RNN-based decoder LMs. Recurrent Neural Networks (RNNs) are neural networks which take a sequential input $y_1, \ldots, y_t$, and produce a sequence of *hidden states* $h_1, \ldots, h_t \in \mathbb{R}^h$. Their defining feature is recurrence – on each step, the RNN applies the same transformation to produce a new hidden state h_t from the previous hidden state h_{t-1} and the current input y_t:

$$h_t = \text{RNN}(h_{t-1}, y_t) \in \mathbb{R}^h \tag{2.4}$$

To produce a probability distribution for the next token y_{t+1}, RNN-based LMs typically apply a *softmax layer* to the hidden state h_t – i.e., a linear transformation followed by the softmax function:

$$s = Wh_t + b \qquad \in \mathbb{R}^V \tag{2.5}$$

$$P_{LM}(y_{t+1}|y_1, \ldots, y_t) = \text{softmax}(s) \qquad \in \mathbb{R}^V \tag{2.6}$$

where $W \in \mathbb{R}^{V \times h}$ and $b \in \mathbb{R}^V$ are trainable parameters, and V is size of the vocabulary $\mathcal{V}$ (see Section 2.4). The softmax function gives the following probability for a word $w \in \mathcal{V}$:

$$P_{LM}(y_{t+1} = w|y_1, \ldots, y_t) = \frac{\exp(s_w)}{\sum_{w' \in \mathcal{V}} \exp(s_{w'})} \tag{2.7}$$

When using a neural LM to provide a representation of y, we can omit the softmax layer and use (some combination of) the hidden states as the representation.

There are several flavors of RNN, which determine the nature of the RNN transformation in Equation 2.4. Simple or 'vanilla' RNNs (Elman, 1990) apply a linear transformation and nonlinearity to h_{t-1} and y_t. One problem with these simple RNNs is the *vanishing gradient problem* – it's difficult for RNNs to learn to preserve information over many timesteps (Bengio et al., 1994). To address this problem, Long Short-Term Memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997; Gers et al., 2000) use both a hidden state h_t and a cell state c_t – which is intended to store long-term information. On each step, the LSTM computes *gates* which determine how the LSTM erases, writes, and reads information from the cell and into the hidden state. This makes it easier for the LSTM to retain information over many timesteps. Gated Recurrent Units (GRUs) (Cho et al., 2014) are a simpler and computationally less expensive variant of LSTMs which do not have a cell state, but also use gates to control read/write behavior. LSTMs and GRUs continue to be the most widely-used RNN variants for neural language models. Other variations are common – for example, bidirectional RNNs compute a backwards sequence of hidden states in addition to the forward sequence computed in Equation 2.4. RNNs are typically *deep* – meaning that successive layers of hidden states are computed using the layers below. For a full exploration of these topics, I refer the reader to Chapter 10 of Goodfellow et al. (2016).

RNN-based encoder-decoder models. Also known as *sequence-to-sequence* models, these models comprise two RNNs: an encoder RNN, which takes a sequential input x, and a decoder RNN, which uses the encoder's representations to compute a probability distribution $P_{LM}(y|x)$ (Sutskever et al., 2014). These RNNs typically have a similar architecture as described in the previous section, though the encoder does not need a softmax layer. Preceding the introduction of attention mechanisms (next section), the final hidden state of the encoder was typically used to represent x; this would be used as the decoder's initial state. However, this introduced a bottleneck problem, as all information about x had to be squeezed into a single fixed-size embedding.

Attention mechanisms. The attention mechanism was devised to ease the bottleneck problem in NMT, while also providing a soft alignment between words in the source sentence x and the target sentence y (Bahdanau et al., 2015). On each step t of the decoder RNN, we take the dot product between the decoder's current hidden state s_t and all of the encoder's hidden states $h_1, \dots, h_n$, then apply the softmax function to obtain the *attention distribution* α:

$$\alpha = \text{softmax}([s_t \cdot h_1, \dots, s_t \cdot h_n]) \in \mathbb{R}^n \tag{2.8}$$

The attention distribution α is a probability distribution over the n source hidden states; it can be regarded as a measure of the relative relevance or usefulness of each source hidden state for the current step t of the decoder. We then use α to compute a weighted sum of the source hidden states; this is the *attention output* a:

$$a = \sum_{i=1}^{n} \alpha_i h_i \in \mathbb{R}^h \tag{2.9}$$

The attention output can be regarded as a summary of the source hidden states, which contains more information from those source hidden states that are more relevant to the current decoder state s_t. The attention output a is then incorporated into the decoder RNN, for example by concatenating with the decoder state s_t. For deep sequence-to-sequence models, the attention mechanism may be applied on each layer of the decoder RNN. Given that sequence-to-sequence models are trained end-to-end (see Section 2.5), the attention mechanism learns to attend to the source states that are most useful to improve the decoder's prediction accuracy. For machine translation, this typically results in an attention distribution that mirrors translation alignment; for tasks like summarization, the model typically attends to the section of the source text being summarized.

Generalizing beyond sequence-to-sequence models, the attention mechanism has become a standard Deep Learning tool: given n *value* vectors $v_1, \dots, v_n \in \mathbb{R}^v$ with corresponding *key* vectors $k_1, \dots, k_n \in \mathbb{R}^k$, and a *query* vector $q \in \mathbb{R}^k$, the attention mechanism computes a weighted sum a of the value vectors, dependent on an attention distribution α computed by taking the dot product between the query and the keys:

$$\alpha = \text{softmax}([q \cdot k_1, \ldots, q \cdot k_n]) \in \mathbb{R}^n \tag{2.10}$$

$$a = \sum_{i=1}^{n} \alpha_i v_i \in \mathbb{R}^v \tag{2.11}$$

There are several alternative formulations of Equation 2.10 (sometimes called *dot product attention*). For example, *multiplicative attention* replaces the dot product $q \cdot k_i$ with a learned weight matrix multiplication qWk_i, while *additive attention* uses a learned additive transformation with nonlinearity $v^\top \tanh(Wq + Uk_i)$ (Luong et al., 2015).

Transformer-based LMs. After the attention mechanism's success across sequence-to-sequence tasks, the Transformer (Vaswani et al., 2017) takes the idea one step further: what if we replace the RNN's recurrent connections with attention everywhere? In a Transformer encoder, each hidden state h_t^l (timestep t, layer l) can attend to all hidden states $h_1^{l-1}, \ldots, h_T^{l-1}$ on the layer below. In a Transformer decoder, h_t^l can attend only to the left $h_1^{l-1}, \ldots, h_t^{l-1}$; this preserves left-to-right causality and permits generation. In addition, the Transformer introduced *multi-head attention*: each hidden state has multiple key, value and query vectors which produce separate attention distributions. This allows the network to learn separate policies of what to attend to with each head.

Since their introduction, Transformers have widely replaced RNNs for all three types of LM described in Section 2.2. Transformers tend to achieve better performance than similarly-sized RNNs; the attention connections seem to allow better and more flexible information flow, especially over long ranges. Transformers' other main advantage over RNNs is encoding efficiency: given an input sequence $x_1, \ldots, x_T$, a Transformer can compute each layer of hidden states in parallel, as each layer $h_1^l, \ldots, h_T^l$ depends only on the previous layer $h_1^{l-1}, \ldots, h_T^{l-1}$. By contrast, an RNN must compute each layer of hidden states left-to-right, as each hidden state h_t^l depends not only on the previous layer h_t^{l-1}, but also on the state to the left h_{t-1}^l. This efficiency advantage does not hold for decoding (i.e., generating) – like a RNN, a Transformer decoder must compute its hidden states left-to-right.

There are some cases where RNNs are preferable to Transformers. First, RNNs can outperform Transformers in low-resource settings (Boito et al., 2019; Araabi and Monz, 2020). Second, the Transformer's self-attention keys and values can introduce an overhead that makes decoding slower than similarly-sized LSTMs; this is why Gmail Smart Compose uses LSTMs in production (Chen et al., 2019). Lastly, Chen et al. (2018) find that some ingredients of the Transformer's success – such as multi-head attention and layer normalization – can be applied along with careful hyperparameter tuning to substantially improve RNN performance. Furthermore they argue that Transformers are generally better for encoding, and recurrent models better for decoding; accordingly they find that a Transformer encoder paired with an improved RNN decoder achieves better MT performance than all other combinations.

2.4 Input Representations for Neural Language Models

Tokenization and word(piece) embeddings. Given some text y, how exactly do we transform it into a sequence of embeddings $y_1, \ldots, y_T \in \mathbb{R}^e$ that can be passed into a neural LM? During the early days of NMT, the standard approach was to first define a fixed *vocabulary* $\mathcal{V}$. For many languages such as English, this would be obtained by taking the training data for that language, separating the text by whitespace (and possibly punctuation) to obtain tokens, then selecting the V most common tokens, where V is some maximum vocabulary size. Each word in the vocabulary would be assigned a *word embedding* – this may be taken from a set of pretrained word embeddings such as word2vec (Mikolov et al., 2013) or GloVe (Pennington et al., 2014), or it may be randomly initialized like other trainable parameters of the NMT model. The primary problem with this approach was that rare words, names, and (depending on the tokenization scheme) hyphenated, compositional, and inflected words are likely to be out-of-vocabulary. Such 'unknown' words were instead represented by an `<UNK>` embedding, meaning that the network receives no information about the identity of the word.

One approach to this problem is to use smaller units of text than words. For example, character-based models (Ling et al., 2015; Luong and Manning, 2016) represent each character in the alphabet with an embedding, then compose these embeddings together to represent larger units of text. However the more successful approach, which is now dominant, is to use *wordpieces*, which are units of text between characters and words. For example, the GPT models (Radford et al., 2018, 2019; Brown et al., 2020) use BPE encoding (Sennrich et al., 2016), which allows the model to encode and generate any Unicode string as a sequence of in-vocabulary tokens; while more common words have their own embeddings, rarer words are split into smaller pieces with their own embeddings. This eliminates usage of the `<UNK>` placeholder and allows models to learn more compositional representations.

Usage of pretrained word embeddings such as word2vec and GloVe has largely been replaced by pretrained LMs. While word2vec and GloVe produce a fixed representation for each word in the vocabulary, LMs take a passage of text and produce a sequence of hidden states that represent each word(piece) *in-context*. For this reason, starting with ELMo (Peters et al., 2018), pretrained LMs' representations are sometimes called *contextual word embeddings*.

Start and end tokens. We usually start and end the target sequence y with special `<START>` and `<END>` tokens which have their own embeddings. The `<START>` token is often needed as the input y_1 for the first timestep of a RNN or Transformer. Meanwhile training the model to predict `<END>` teaches it when to stop generating; the token is an important stopping condition for decoding algorithms (Section 2.6).

Framing conditional generative tasks as continuation. Large-scale generative pretraining – for example, the GPT models (Radford et al., 2018, 2019; Brown et al., 2020) – is based on unsupervised learning $P_{LM}(y)$ of unlabelled internet text y. However, these pretrained models can be used for conditional generative tasks $P_{LM}(y|x)$ if x can be represented as text. This is typically achieved by concatenating x and y (perhaps separated by a separator token `<SEP>`) to form a single sequence, and framing the task as continuation $P_{LM}(y_{t+1}, \ldots, y_T|y_1, \ldots, y_t)$. Similarly, if x is multiple pieces of text (e.g., a dialogue history), it can be concatenated into a single sequence (for dialogue, perhaps including the speaker's identity for each utterance). More generally, there is a trend towards encoding task meta-data – for example, the desired language or style of the output text, or even the task to be performed – as part of y (McCann et al., 2018; Keskar et al., 2019; Radford et al., 2019; Brown et al., 2020). This makes convenient use of a pretrained $P_{LM}(y)$ model without needing to train separate models or add new special tokens.

2.5 Training and Evaluating Neural Language Models

Maximum likelihood estimation training. Neural language models are typically trained end-to-end with backpropagation and stochastic gradient descent. The *maximum likelihood estimation* (MLE) training objective is the *negative log likelihood* of the target data according to P_{LM}; this is equivalent to the *cross-entropy loss* between the data distribution and P_{LM}. More concretely, given training examples $(x, y) \in \mathcal{D}_{\text{train}}$, the cross-entropy loss for an encoder-decoder LM is:

$$CE(\mathcal{D}_{\text{train}}) = -\frac{1}{N_{\text{train}}} \sum_{(x,y)\in\mathcal{D}_{\text{train}}} \log P_{LM}(y|x) \tag{2.12}$$

where N_{train} is the total number of tokens in the output sequences y. For an unconditional decoder LM that models $P_{LM}(y)$, simply omit x from Equations 2.12 and 2.13.

Perplexity evaluation. Neural language models are typically evaluated by *perplexity* – given test examples $(x, y) \in \mathcal{D}_{\text{test}}$, this is:

$$PP(\mathcal{D}_{\text{test}}) = \left(\prod_{(x,y)\in\mathcal{D}_{\text{test}}} P_{LM}(y|x) \right)^{-1/N_{\text{test}}} \tag{2.13}$$

This is the inverse P_{LM} probability of the test set y given x, normalized by N_{test}, the total number of tokens in the output sequences y. Perplexity is equal to the exponential of the cross-entropy loss; thus there is a close relationship between the MLE training objective and the perplexity evaluation objective.

Teacher forcing and exposure bias. During training with MLE, the LM learns to predict each next token y_t conditioned on the preceding tokens $y_1, \ldots, y_{t-1}$ taken from the training data; by contrast during generation the preceding tokens have been generated by the LM itself (Section 2.6). This training regime is called *teacher forcing*, because the LM is being forced to follow the sequence provided by the training data (the teacher), whether or not the LM agrees with that sequence. Teacher forcing is thought to cause a domain shift problem, as during training the LM is only exposed to training text and not its own generated text – which may differ distributionally. This *exposure bias* is thought to cause generation problems, in particular an inability to recover from errors.

Reinforcement learning. One solution to exposure bias is to use *reinforcement learning* (RL) to optimize the LM for a sequence-level reward (e.g., BLEU or ROUGE), measured with respect to the LM-generated text, during training (Ranzato et al., 2016). Reward can also be awarded on intermediate steps of the sequence (Bahdanau et al., 2017). In their survey of RL-based methods; Keneshloo et al. (2019) note that training text generation models with RL is generally harder than with MLE, due to issues including sample efficiency, high variance and the large action space. They also note that directly optimizing for metrics such as BLEU or ROUGE can be problematic as those scores are not good representatives of human-judged quality; this has been found in practical applications (Wu et al., 2016). In most cases, Keneshloo et al. (2019) find that RL-based methods do not currently provide a significant improvement over MLE.

Adversarial training. Another solution to exposure bias is to use *generative adversarial networks* (GANs) (Goodfellow et al., 2014) – a method originally devised for image generation, in which a discriminator network learns to distinguish between real images and model-generated images, and the generator network learns to fool the discriminator. GANs have been applied to text generation – for example Lamb et al. (2016) propose *professor forcing*, in which the discriminator network takes the outputs and hidden states of the generator RNN, and learns to distinguish whether the RNN is running in self-generating or teacher-forcing mode. Discriminators can be used to target particular desirable aspects of text; Holtzman et al. (2018) provide particular features of the generator RNN to their discriminators, in order to guide each to focus on one of Grice's maxims (Grice, 1975). Overall, text GANs have not achieved the success of image GANs, nor have they surpassed MLE in performance, primarily because they are hindered by training issues such as mode collapse, vanishing gradient, and the non-differentiability of the text output (Caccia et al., 2018; Iqbal and Qureshi, 2020).

Variational autoencoders. Another approach to text generation is *variational autoencoders* (VAEs) (Kingma and Welling, 2014). In an *autoencoder*, an encoder produces a latent vector representation of the input text, which the decoder uses to reproduce the original. In a *variational* autoencoder, the latent representation is *sampled* from the latent space, using a mean and variance supplied by the encoder. This provides a smooth latent space of text representations, which can be used to sample text (Bowman et al., 2016). This line of work has been useful for stylized text generation and style transfer (Mou and Vechtomova, 2020). However, this approach is also held back by training difficulties – in particular, *KL collapse*, in which the decoder ignores the latent representation (Iqbal and Qureshi, 2020).

2.6 Decoding Algorithms

Once we have a neural LM that models $P_{LM}(y)$ or $P_{LM}(y|x)$, how do we generate the output text y?

2.6.1 Likelihood-maximizing Decoding Algorithms

We may be interested in finding the highest probability y – that is, the sequence y which maximizes $P_{LM}(y|x)$. Likelihood-maximizing decoding algorithms aim to fulfil this goal.

Greedy decoding. As the name suggests, greedy decoding is a simple approach to finding a high-probability sequence by greedily choosing the highest-probability token on each step. That is, having generated $y_1, ..., y_{t-1}$, greedy decoding chooses

$$y_t = \text{argmax}_{y_t \in \mathcal{V}} P_{LM}(y_t|y_1, ..., y_{t-1}, x) \tag{2.14}$$

then proceeds to the next step. Greedy decoding typically stops when an `<END>` token is generated, or when a maximum length has been reached.

The most obvious problem with greedy decoding is that it is not guaranteed to find the highest-probability sequence y. In particular, the optimal y may lie on a path that requires

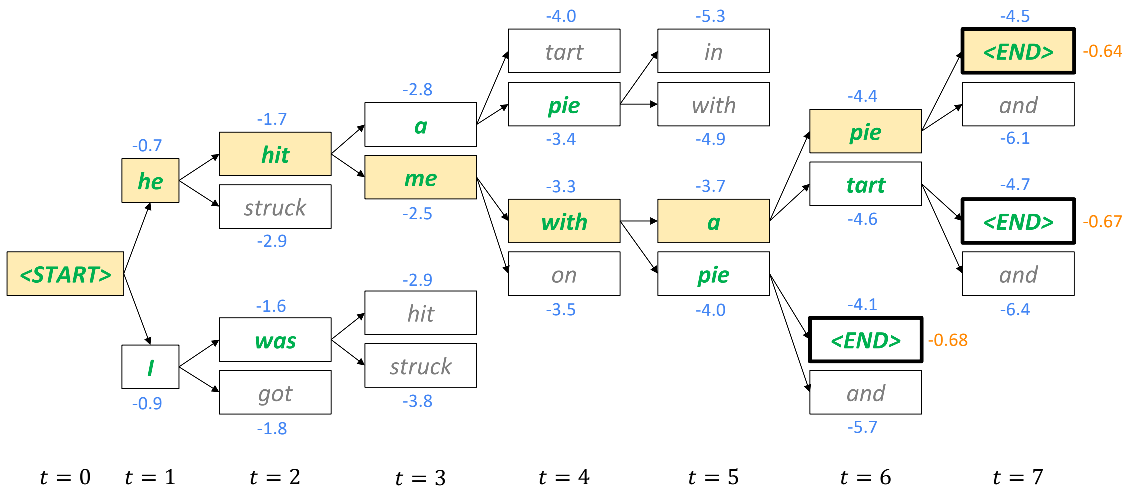


Figure 2.1: Beam search example with beam size 2 , showing unnormalized scores (Equation 2.15) in blue, and length-normalized scores (Equation 2.16) in orange for the finished hypotheses (bold boxes). The finished sequence with highest length-normalized score is shaded orange. Best viewed in color. Adapted from Lecture 8 of CS224n 2019 (Manning and See, 2019).

choosing some y_t which is not the most probable token on that step. Greedy decoding explores only one path with no backtracking, so it will not necessarily discover the optimal y.

Beam search decoding. Given that the space of outputs of length T is of size $O(V^T)$ where V is vocabulary size, it is not possible to exhaustively search for the optimal solution. Beam search is a balance between greedy and exhaustive search – while not exploring the whole solution space, beam search explores more solutions than greedy decoding. To achieve this, beam search continually maintains and updates a set of b partially-generated solutions, sometimes called *hypotheses*. This set of hypotheses is called the *beam*; it represents the b highest-probability sequences encountered so far. The parameter b is called the *beam size*, and can be chosen to tradeoff between lower computational complexity (small b) and more thorough search (large b). In fact, $b = 1$ is equivalent to greedy search, and $b = V$ to exhaustive search.

For example, Figure 2.1 shows beam search decoding an English translation of the

French sentence *il m'a entarté* with $b = 2$. On step $t = 0$, the beam (represented by bold green text) contains one hypothesis: the `<START>` token. On step $t = 1$, the beam contains the top b most likely next tokens following `<START>`, which are *he* and *I*. The blue number next to each box represents the *score* of that hypothesis, which is its log probability:

$$\text{score}(y_1, \ldots, y_t) = \sum_{i=1}^{t} P_{LM}(y_i|y_1, \ldots, y_{i-1}, x) \tag{2.15}$$

On each step $t \geq 2$, we take the beam from the previous step (bold green text), and for each hypothesis, consider the top b most likely next tokens. From this set of b^2 hypotheses we keep only the b most likely (i.e., with highest blue score numbers) – this is our new beam.[1]

We regard a hypothesis as *finished* if it ends in the `<END>` token. Hypotheses may finish on different timesteps – in Figure 2.1, there is one finished hypothesis on $t = 6$ and two on $t = 7$ (indicated by bold boxes). When the beam contains a finished hypothesis (e.g., on step $t = 6$), we remove it from the beam, place it aside in a set of finished solutions, and continue the search. Typically, beam search terminates when a desired number of finished solutions have been found (commonly b) – under this condition, beam search would stop on step $t = 7$. Otherwise, it may stop upon reaching some maximum length.

Finally, we have a set of finished solutions, which may be of varying length. It may be intuitive to choose the highest-scoring (i.e., most probable) solution – this would be `<START>` *he hit me with pie* `<END>` with score -4.1. However, this demonstrates a problem: as a sequence lengthens, its overall probability decreases. Thus, the highest-scoring solution typically biases towards shorter sequences that are not necessarily better – in this example, omitting the word *a* is unnatural. The typical solution to this problem is *length normalization* which replaces Equation 2.15 with:

$$\text{score}(y_1, \ldots, y_t) = \frac{1}{t} \sum_{i=1}^{t} P_{LM}(y_i|y_1, \ldots, y_{i-1}, x) \tag{2.16}$$

This adjustment emphasizes the *average* probability of tokens in the sequence, rather than their cumulative probability. With this adjustment, `<START>` *he hit me with a pie* `<END>`

[1] Although we choose our new set of hypotheses to be the b most likely of these b^2 possible continuations, it is logically equivalent to selecting the b most likely from all bV possible continuations.

is the solution with highest normalized score of -0.64. Some researchers have found it useful to replace $1/t$ in Equation 2.16 with $1/t^{\alpha}$, where $0 < \alpha < 1$ is a tunable *length normalization penalty* that controls the strength of the length normalization (Wu et al., 2016).

2.6.2 Sampling-based Decoding Algorithms

Instead of searching for the highest probability y, we may instead wish to use $P_{LM}(y|x)$, or a modification of it, to randomly *sample* the output y. Sampling-based decoding algorithms take this approach.

Natural sampling. This involves sampling directly from the natural distribution provided by $P_{LM}(y|x)$. To achieve this, we simply sample each successive token from P_{LM}:

$$y_t \sim P_{LM}(y_t|y_1, ..., y_{t-1}, x) \tag{2.17}$$

We may stop generating either when the sequence produces an `<END>` token, or when some maximum length is reached. The resulting sequence y is a true sample from the underlying distribution P_{LM}. In practice, the LM tends to put enough probability on unsuitable tokens that after some number of generation steps, a bad token will be chosen; after this it can be difficult to recover. Consequently, natural sampling tends to produce syntactically fluent but nonsensical text, even for large pretrained LMs (Holtzman et al., 2020).

Top-k sampling. The main idea of top-k sampling (Fan et al., 2018b) is to sample each successive token from the k most likely options on that step. The parameter k can be chosen to tradeoff between higher-probability, generic sequences (low k) and more varied, unusual sequences (high k).

Formally: given a partial sequence $y_1, \ldots, y_{t-1}$, let S_k contain the k tokens $y_t \in \mathcal{V}$ with largest $P_{LM}(y_t|y_1, ..., y_{t-1}, x)$. We then define the truncated distribution $P_{\text{top}-k}$:

$$P_{\text{top}-k}(y_t|y_1, ..., y_{t-1}, x) = \begin{cases} \frac{1}{Z} P_{LM}(y_t|y_1, ..., y_{t-1}, x), & \text{if } y_t \in S_k \\ 0, & \text{otherwise} \end{cases} \tag{2.18}$$

where Z is a normalizing constant: $Z = \sum_{y_t \in S_k} P_{LM}(y_t|y_1, ..., y_{t-1}, x)$. Top-$k$ sampling generates text by successively sampling each y_t from the truncated $P_{\text{top}-k}$ distribution:

$$y_t \sim P_{\text{top}-k}(y_t|y_1, ..., y_{t-1}, x) \tag{2.19}$$

Top-k sampling can be seen as a balance between greedy decoding ($k = 1$) and natural sampling ($k = V$).

Top-p, aka nucleus sampling. One argument against top-k sampling is that k should fluctuate from step to step. For example, when P_{LM} is a relatively flat distribution, spreading its probably mass over many likely tokens, we may prefer a larger k so we can include those tokens. By contrast, when P_{LM} is peaked, placing much of its probability mass on one or a few tokens, we may prefer a smaller k so we only consider the tokens the model is sure about.

This is the idea of top-p (also called *nucleus*) sampling (Holtzman et al., 2020) – instead of sampling from a constant number k of most likely tokens, we set a probability p and on each step sample from whatever number of tokens together constitute the 'top' p of the probability distribution. Holtzman et al. call this set the *nucleus*. Formally: given a probability $0 \leq p \leq 1$ and a partial sequence $y_1, \ldots, y_{t-1}$, we define the truncated distribution $P_{\text{top}-p}$:

$$P_{\text{top}-p}(y_t|y_1, ..., y_{t-1}, x) = P_{\text{top}-k}(y_t|y_1, ..., y_{t-1}, x) \tag{2.20}$$

$$\text{where } k \text{ min s.t.} \sum_{y_t \in S_k} P_{LM}(y_t|y_1, ..., y_{t-1}, x) \geq p \tag{2.21}$$

Like top-k sampling, the algorithm proceeds by successively sampling each y_t from $P_{\text{top}-p}$. Top-p sampling is equivalent to greedy decoding when $p = 0$ and natural sampling when $p = 1$.

2.6.3 Temperature

Another way to adjust the genericness of the generated text is the *temperature parameter* $0 < \tau < \infty$; this transforms Equation 2.7 to:

$$P_{LM}(y_{t+1} = w|y_1, \ldots, y_t) = \frac{\exp(s_w/\tau)}{\sum_{w' \in \mathcal{V}} \exp(s_{w'}/\tau)} \tag{2.22}$$

Lowering the temperature below 1 makes P_{LM} more peaked, concentrating the probability on the most likely words. Conversely, raising the temperature above 1 makes P_{LM} more flat, spreading the probability around the vocabulary. Temperature can be used in conjunction with any of the decoding algorithms presented in this section.

Chapter 3

Summarization with Pointer-Generator Networks

3.1 Introduction

This chapter presents a method for abstractive summarization of long text. While the input text provides a strong constraint on the output summary, this task is still much more open-ended than Machine Translation (Figure 1.1). At the time of this work, researchers had only recently begun to apply the NMT blueprint (Section 1.2) to more open-ended tasks; thus this chapter focuses on understanding the different requirements of abstractive summarization, and adapting the NMT blueprint to better serve those requirements.

Summarization is the task of condensing a piece of text to a shorter version that contains the main information from the original. There are two broad approaches to summarization: *extractive* and *abstractive*. *Extractive methods* assemble summaries exclusively from passages (usually whole sentences) taken directly from the source text, while *abstractive methods* may generate novel words and phrases not featured in the source text – as a human-written abstract usually does. The extractive approach is easier, because copying large chunks of text from the source document ensures baseline levels of grammaticality and accuracy. On the other hand, sophisticated abilities that are crucial to high-quality summarization, such as paraphrasing, generalization, or the incorporation of real-world knowledge, are possible only in an abstractive framework (see Figure 3.5).

Original Text (truncated): lagos, nigeria (cnn) a day after winning nigeria's presidency, ***muhammadu buhari*** told cnn's christiane amanpour that **he plans to aggressively fight corruption that has long plagued nigeria** and go after the root of the nation's unrest. *buhari* said he'll "rapidly give attention" to curbing violence in the northeast part of nigeria, where the terrorist group boko haram operates. by cooperating with neighboring nations chad, cameroon and niger, **he said his administration is confident it will be able to thwart criminals** and others contributing to nigeria's instability. for the first time in nigeria's history, the opposition defeated the ruling party in democratic elections. *buhari* defeated incumbent goodluck jonathan by about 2 million votes, according to nigeria's independent national electoral commission. **the win comes after a long history of military rule, coups and botched attempts at democracy in africa's most populous nation.**

Baseline Seq2Seq + Attention: UNK UNK says his administration is confident it will be able to **destabilize nigeria's economy**. UNK says his administration is confident it will be able to thwart criminals and other **nigerians**. **he says the country has long nigeria and nigeria's economy.**

Pointer-Gen: *muhammadu buhari* says he plans to aggressively fight corruption **in the northeast part of nigeria**. he says he'll "rapidly give attention" to curbing violence **in the northeast part of nigeria**. he says his administration is confident it will be able to thwart criminals.

Pointer-Gen + Coverage: *muhammadu buhari* says he plans to aggressively fight corruption that has long plagued nigeria. he says his administration is confident it will be able to thwart criminals. the win comes after a long history of military rule, coups and botched attempts at democracy in africa's most populous nation.

Figure 3.1: Comparison of 3 abstractive summarization models on a news article. The baseline model makes **factual errors**, a **nonsensical sentence** and struggles with OOV words *muhammadu buhari*. The pointer-generator model is accurate but **repeats itself**. Coverage eliminates repetition. The final summary is composed from **several fragments**.

Due to the difficulty of abstractive summarization, the majority of past work has been extractive (Kupiec et al., 1995; Paice, 1990; Nenkova and McKeown, 2011; Saggion and Poibeau, 2013). However, the success of *sequence-to-sequence* models (Sutskever et al., 2014), in which recurrent neural networks (RNNs) both read and freely generate text, has made abstractive summarization increasingly viable (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016; Zeng et al., 2016). However, these systems exhibit undesirable behavior such as inaccurately reproducing factual details, an inability to deal with out-of-vocabulary (OOV) words, and repeating themselves (see Figure 3.1).

In this chapter we present an architecture that addresses these three issues in the context of multi-sentence summaries. While most prior abstractive work focused on headline generation tasks (reducing one or two sentences to a single headline), we believe that longer-text summarization is both more challenging (requiring higher levels of abstraction while avoiding repetition) and ultimately more useful. Therefore we apply our model to the *CNN/Daily Mail* dataset (Hermann et al., 2015; Nallapati et al., 2016), which contains news articles (39 sentences on average) paired with multi-sentence summaries, and show that our model outperforms the prior state-of-the-art abstractive system by at least 2 ROUGE points.

Our hybrid *pointer-generator* network facilitates copying words from the source text via *pointing* (Vinyals et al., 2015a), which improves accuracy and handling of OOV words, while retaining the ability to *generate* new words. The network, which can be viewed as a balance between extractive and abstractive approaches, is similar to CopyNet (Gu et al., 2016) and Forced-Attention Sentence Compression (Miao and Blunsom, 2016), that were applied to short-text summarization. We propose a novel variant of the *coverage vector* (Tu et al., 2016) from Neural Machine Translation, which we use to track and control coverage of the source document. We show that coverage is remarkably effective for eliminating repetition.

3.2 Our Models

In this section we describe (1) our baseline sequence-to-sequence model, (2) our pointer-generator model, and (3) our coverage mechanism that can be added to either of the first two models. The code for our models is available online.[1]

3.2.1 Sequence-to-sequence Attentional Model

Our baseline model is similar to that of Nallapati et al. (2016), and is depicted in Figure 3.2. The tokens of the article w_i are fed one-by-one into the encoder (a single-layer bidirectional LSTM), producing a sequence of *encoder hidden states* h_i. On each step t, the decoder (a single-layer unidirectional LSTM) receives the word embedding of the previous word (while training, this is the previous word of the reference summary; at test time it is the previous word emitted by the decoder), and has *decoder state* s_t. The *attention distribution* α^t is calculated as in Bahdanau et al. (2015)[2]:

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + b_{\text{attn}}) \tag{3.1}$$

$$\alpha^t = \text{softmax}(e^t) \tag{3.2}$$

[1] www.github.com/abisee/pointer-generator

[2] This is *additive attention* as mentioned in Section 2.3.

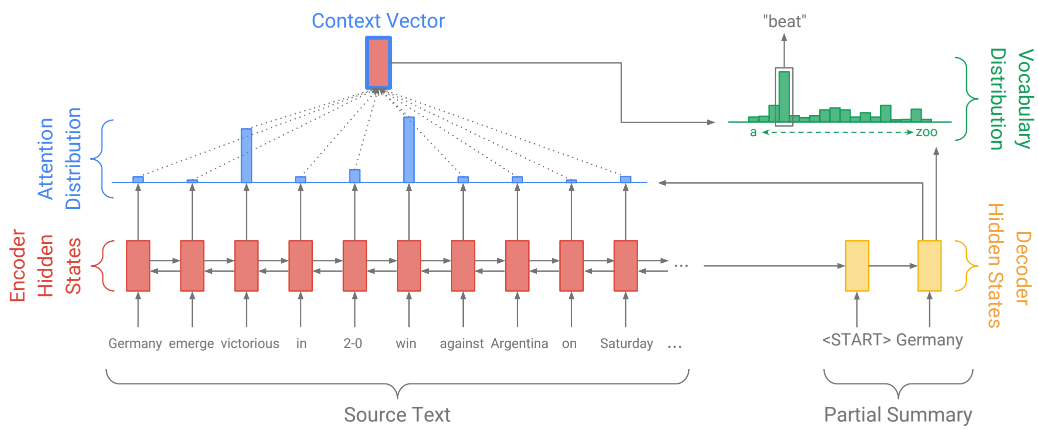


Figure 3.2: Baseline sequence-to-sequence model with attention. The model may attend to relevant words in the source text to generate novel words, e.g., to produce the novel word *beat* in the abstractive summary *Germany* ***beat*** *Argentina 2-0* the model may attend to the words *victorious* and *win* in the source text.

where v, W_h, W_s and b_{attn} are learnable parameters. As discussed in Chapter 2, the attention distribution can be viewed as a probability distribution over the source words, that tells the decoder where to look to produce the next word. Next, the attention distribution is used to produce a weighted sum of the encoder hidden states, known as the *context vector*[3] h_t^*:

$$h_t^* = \sum_i \alpha_i^t h_i \tag{3.3}$$

The context vector, which can be seen as a fixed-size representation of what has been read from the source for this step, is concatenated with the decoder state s_t and fed through two linear layers to produce the vocabulary distribution P_{vocab}:

$$P_{\text{vocab}} = \text{softmax}(V'(V[s_t, h_t^*] + b) + b') \tag{3.4}$$

where V, V', b and b' are learnable parameters. P_{vocab} is a probability distribution over all words in the vocabulary, and provides us with our final distribution from which to predict words w:

$$P(w) = P_{\text{vocab}}(w) \tag{3.5}$$

[3]Sometimes also called the *attention output* a_t, as in Section 2.3

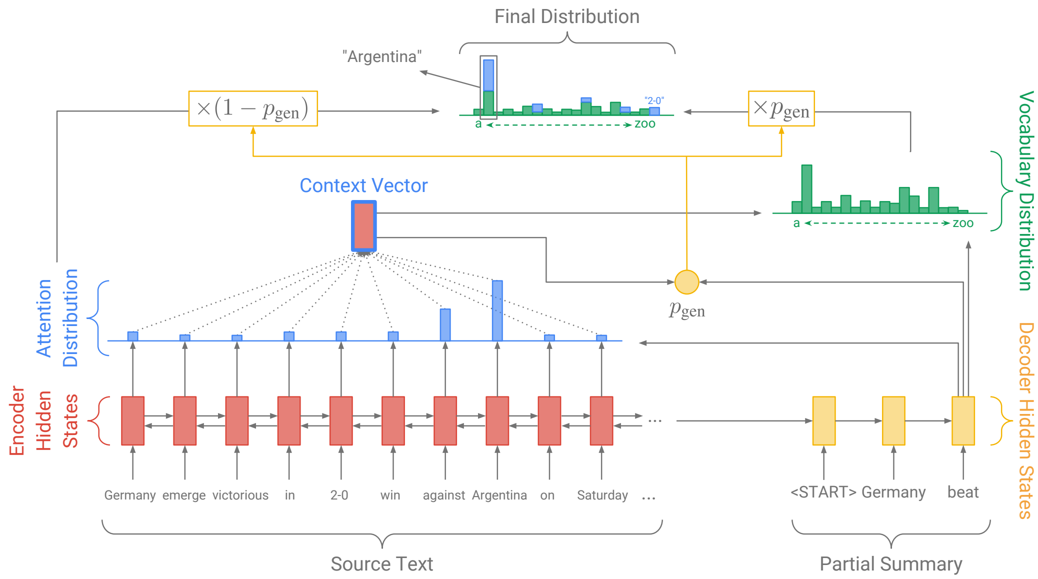


Figure 3.3: Pointer-generator model. For each decoder timestep a generation probability $p_{\text{gen}} \in [0, 1]$ is calculated, which weights the probability of *generating* words from the vocabulary, versus *copying* words from the source text. The vocabulary distribution and the attention distribution are weighted and summed to obtain the final distribution, from which we make our prediction. Note that out-of-vocabulary article words such as *2-0* are included in the final distribution. Best viewed in color.

During training, the loss for timestep t is the negative log likelihood of the target word w_t^* for that timestep:

$$\text{loss}_t = -\log P(w_t^*) \tag{3.6}$$

and the overall loss for the whole sequence is:

$$\text{loss} = \frac{1}{T} \sum_{t=0}^{T} \text{loss}_t \tag{3.7}$$

3.2.2 Pointer-generator Network

Our pointer-generator network is a hybrid between our baseline and a pointer network (Vinyals et al., 2015a), as it allows both copying words via pointing, and generating words from a fixed vocabulary. In the pointer-generator model (depicted in Figure 3.3) the attention distribution α^t and context vector h_t^* are calculated as in section 3.2.1. In addition, the *generation probability* $p_{\text{gen}} \in [0, 1]$ for timestep t is calculated from the context vector h_t^*, the decoder state s_t and the decoder input x_t:

$$p_{\text{gen}} = \sigma(w_{h^*}^T h_t^* + w_s^T s_t + w_x^T x_t + b_{\text{ptr}}) \tag{3.8}$$

where vectors w_{h^*}, w_s, w_x and scalar b_{ptr} are learnable parameters and σ is the sigmoid function. Next, p_{gen} is used as a soft switch to choose between *generating* a word from the vocabulary by sampling from P_{vocab}, or *copying* a word from the input sequence by sampling from the attention distribution α^t. For each document let the *extended vocabulary* denote the union of the vocabulary, and all words appearing in the source document. We obtain the following probability distribution over the extended vocabulary:

$$P(w) = p_{\text{gen}} P_{\text{vocab}}(w) + (1 - p_{\text{gen}}) \sum_{i:w_i=w} \alpha_i^t \tag{3.9}$$

Note that if w is an out-of-vocabulary (OOV) word, then $P_{\text{vocab}}(w)$ is zero; similarly if w does not appear in the source document, then $\sum_{i:w_i=w} \alpha_i^t$ is zero. The ability to produce OOV words is one of the primary advantages of pointer-generator models; by contrast models such as our baseline are restricted to their pre-set vocabulary.

The loss function is as described in equations (3.6) and (3.7), but with respect to our modified probability distribution $P(w)$ given in equation (3.9).

3.2.3 Coverage Mechanism

Repetition is a common problem for sequence-to-sequence models (Tu et al., 2016; Mi et al., 2016; Sankaran et al., 2016; Suzuki and Nagata, 2017), and is especially pronounced when generating multi-sentence text (see Figure 3.1). We adapt the *coverage model* of Tu et al. (2016) to solve the problem. In our coverage model, we maintain a *coverage vector* c^t, which is the sum of attention distributions over all previous decoder timesteps:

$$c^t = \sum_{t'=0}^{t-1} \alpha^{t'} \tag{3.10}$$

Intuitively, c^t is a (unnormalized) distribution over the source document words that represents the degree of coverage that those words have received from the attention mechanism so far. Note that c^0 is a zero vector, because on the first timestep, none of the source document has been covered.

The coverage vector is used as extra input to the attention mechanism, changing equation (3.1) to:

$$e_i^t = v^T \tanh(W_h h_i + W_s s_t + w_c c_i^t + b_{\text{attn}}) \tag{3.11}$$

where w_c is a learnable parameter vector of the same length as v. This ensures that the attention mechanism's current decision (choosing where to attend next) is informed by a reminder of its previous decisions (summarized in c^t). This should make it easier for the attention mechanism to avoid repeatedly attending to the same locations, and thus avoid generating repetitive text.

We find it necessary (see section 3.5) to additionally define a *coverage loss* to penalize repeatedly attending to the same locations:

$$\text{covloss}_t = \sum_i \min(\alpha_i^t, c_i^t) \tag{3.12}$$

Note that the coverage loss is bounded; in particular $\text{covloss}_t \leq \sum_i \alpha_i^t = 1$.

Equation (3.12) differs from the coverage loss used in Machine Translation. In MT, we assume that there should be a roughly one-to-one translation ratio; accordingly the final coverage vector is penalized if it is more or less than 1. Our loss function is more flexible: because summarization should not require uniform coverage, we only penalize the overlap between each attention distribution and the coverage so far – preventing repeated attention. Finally, the coverage loss, reweighted by some hyperparameter λ, is added to the primary loss function to yield a new composite loss function:

$$\text{loss}_t = -\log P(w_t^*) + \lambda \sum_i \min(\alpha_i^t, c_i^t) \tag{3.13}$$

3.3 Related Work

Neural abstractive summarization. Rush et al. (2015) were the first to apply modern neural networks to abstractive text summarization, achieving state-of-the-art performance on DUC-2004 and Gigaword, two sentence-level summarization datasets. Their approach, which is centered on the attention mechanism, has been augmented with recurrent decoders (Chopra et al., 2016), Abstract Meaning Representations (Takase et al., 2016), hierarchical networks (Nallapati et al., 2016), variational autoencoders (Miao and Blunsom, 2016), and direct optimization of the performance metric (Ranzato et al., 2016), further improving performance on those datasets.

However, large-scale datasets for summarization of *longer* text are rare. Nallapati et al. (2016) adapted the DeepMind question-answering dataset (Hermann et al., 2015) for summarization, resulting in the *CNN/Daily Mail* dataset, and provided the first abstractive baselines. The same authors then published a neural *extractive* approach (Nallapati et al., 2017), which uses hierarchical RNNs to select sentences, and found that it significantly outperformed their abstractive result with respect to the ROUGE metric. To our knowledge, these were the only two published results on the full dataset at the time of our work.

Prior to modern neural methods, abstractive summarization received less attention than extractive summarization, but Jing (2000) explored cutting unimportant parts of sentences to create summaries, and Cheung and Penn (2014) explore sentence fusion using dependency trees.

Pointer-generator networks. The pointer network (Vinyals et al., 2015a) is a sequence-to-sequence model that uses the soft attention distribution of Bahdanau et al. (2015) to produce an output sequence consisting of elements from the input sequence. The pointer network has been used to create hybrid approaches for NMT (Gulcehre et al., 2016), language modeling (Merity et al., 2016), and summarization (Gu et al., 2016; Gulcehre et al., 2016; Miao and Blunsom, 2016; Nallapati et al., 2016; Zeng et al., 2016).

Our approach is close to the Forced-Attention Sentence Compression model of Miao and Blunsom (2016) and the CopyNet model of Gu et al. (2016), with some small differences: (i) We calculate an explicit switch probability p_{gen}, whereas Gu et al. induce competition through a shared softmax function. (ii) We recycle the attention distribution to serve as the copy distribution, but Gu et al. use two separate distributions. (iii) When a word appears multiple times in the source text, we sum probability mass from all corresponding parts of the attention distribution, whereas Miao and Blunsom do not. Our reasoning is that (i) calculating an explicit p_{gen} usefully enables us to raise or lower the probability of all generated words or all copy words at once, rather than individually, (ii) the two distributions serve such similar purposes that we find our simpler approach suffices, and (iii) we observe that the pointer mechanism often copies a word while attending to multiple occurrences of it in the source text.

Our approach is considerably different from that of Gulcehre et al. (2016) and Nallapati et al. (2016). Those works train their pointer components to activate only for out-of-vocabulary words or named entities (whereas we allow our model to freely learn when to use the pointer), and they do not mix the probabilities from the copy distribution and the vocabulary distribution. We believe the mixture approach described here is better for abstractive summarization – in section 3.6 we show that the copy mechanism is vital for accurately reproducing rare but in-vocabulary words, and in section 3.7.2 we observe that the mixture model enables the language model and copy mechanism to work together to perform abstractive copying.

Coverage. Originating from Statistical Machine Translation (Koehn, 2009), coverage was adapted for NMT by Tu et al. (2016) and Mi et al. (2016), who both use a GRU to update the coverage vector each step. We find that a simpler approach – summing the attention distributions to obtain the coverage vector – suffices. In this respect our approach is similar to Xu et al. (2015), who apply a coverage-like method to image captioning, and Chen et al. (2016), who also incorporate a coverage mechanism (which they call 'distraction') as described in equation (3.11) into neural summarization of longer text.

Temporal attention is a related technique that has been applied to NMT (Sankaran et al., 2016) and summarization (Nallapati et al., 2016). In this approach, each attention distribution is divided by the sum of the previous, which effectively dampens repeated attention. We tried this method but found it too destructive, distorting the signal from the attention mechanism and reducing performance. We hypothesize that an early intervention method such as coverage is preferable to a post hoc method such as temporal attention – it is better to *inform* the attention mechanism to help it make better decisions, than to *override* its decisions altogether. This theory is supported by the large boost that coverage gives our ROUGE scores (see Table 3.1), compared to the smaller boost given by temporal attention for the same task (Nallapati et al., 2016).

3.4 Dataset

We use the *CNN/Daily Mail* dataset (Hermann et al., 2015; Nallapati et al., 2016), which contains online news articles (781 tokens on average) paired with multi-sentence summaries (3.75 sentences or 56 tokens on average). We used scripts supplied by Nallapati et al. (2016) to obtain the same version of the the data, which has 287,226 training pairs, 13,368 validation pairs and 11,490 test pairs. At the time of our work, Nallapati et al.'s published results (Nallapati et al., 2016, 2017) used the *anonymized* version of the data, which has been pre-processed to replace each named entity, e.g., *The United Nations*, with its own unique identifier for the example pair, e.g., `@entity5`. By contrast, we operate directly on the original text (or *non-anonymized* version of the data),[4] which we believe is the favorable problem to solve because it requires no pre-processing.

3.5 Experiments

For all experiments, our model has 256-dimensional hidden states and 128-dimensional word embeddings. For the pointer-generator models, we use a vocabulary of 50k words for both source and target – note that due to the pointer network's ability to handle OOV words, we can use a smaller vocabulary size than the 150k source and 60k target vocabularies in Nallapati et al. (2016). For the baseline model, we also try a larger vocabulary size of 150k.

Note that the pointer and the coverage mechanism introduce very few additional parameters to the network: for the models with vocabulary size 50k, the baseline model has 21,499,600 parameters, the pointer-generator adds 1153 extra parameters (w_{h^*}, w_s, w_x and b_{ptr} in equation 3.8), and coverage adds 512 extra parameters (w_c in equation 3.11).

Unlike Nallapati et al. (2016), we do not pre-train the word embeddings – they are learned from scratch during training. We train using Adagrad (Duchi et al., 2011) with

[4]Instructions to obtain data at `www.github.com/abisee/pointer-generator`

learning rate 0.15 and an initial accumulator value of 0.1. (This was found to work best of Stochastic Gradient Descent, Adadelta, Momentum, Adam and RMSProp). We use gradient clipping with a maximum gradient norm of 2, but do not use any form of regularization. We use loss on the validation set to implement early stopping.

During training and at test time we truncate the article to 400 tokens and limit the length of the summary to 100 tokens for training and 120 tokens at test time.[5] This is done to expedite training and testing, but we also found that truncating the article can *raise* the performance of the model (see section 3.7.1 for more details). For training, we found it efficient to start with highly-truncated sequences, then raise the maximum length once converged. We train on a single Tesla K40m GPU with a batch size of 16. At test time our summaries are produced using beam search with beam size 4.

We trained both our baseline models for about 600,000 iterations (33 epochs) – this is similar to the 35 epochs required by Nallapati et al.'s (2016) best model. Training took 4 days and 14 hours for the 50k vocabulary model, and 8 days 21 hours for the 150k vocabulary model. We found the pointer-generator model quicker to train, requiring less than 230,000 training iterations (12.8 epochs); a total of 3 days and 4 hours. In particular, the pointer-generator model makes much quicker progress in the early phases of training. To obtain our final coverage model, we added the coverage mechanism with coverage loss weighted to $\lambda = 1$ (as described in equation 3.13), and trained for a further 3000 iterations (about 2 hours). In this time the coverage loss converged to about 0.2, down from an initial value of about 0.5. We also tried a more aggressive value of $\lambda = 2$; this reduced coverage loss but increased the primary loss function, thus we did not use it.

We tried training the coverage model without the loss function, hoping that the attention mechanism may learn by itself not to attend repeatedly to the same locations, but we found this to be ineffective, with no discernible reduction in repetition. We also tried training with coverage from the first iteration rather than as a separate training phase, but found that in the early phase of training, the coverage objective interfered with the main objective, reducing overall performance.

[5] The upper limit of 120 is mostly invisible: the beam search algorithm is self-stopping and almost never reaches the 120th step.

	ROUGE			METEOR	
	1	2	L	exact match	+ stem/synonym/ paraphrase
abstr. model (Nallapati et al., 2016)*	35.46	13.30	32.65	-	-
seq-to-seq + attn baseline (150k vocab)	30.49	11.17	28.08	11.65	12.86
seq-to-seq + attn baseline (50k vocab)	31.33	11.81	28.83	12.03	13.20
pointer-generator	36.44	15.66	33.42	15.35	16.65
pointer-generator + coverage	**39.53**	**17.28**	**36.38**	17.32	18.72
lead-3 baseline (ours)	40.34	17.70	36.57	20.48	22.21
lead-3 baseline (Nallapati et al., 2017)*	39.2	15.7	35.5	-	-
extr. model (Nallapati et al., 2017)*	39.6	16.2	35.3	-	-

Table 3.1: ROUGE F_1 and METEOR scores on the test set. Models and baselines in the top half are abstractive, while those in the bottom half are extractive. Those marked with * were trained and evaluated on the anonymized dataset, and so are not strictly comparable to our results on the original text. All our ROUGE scores have a 95% confidence interval of at most ± 0.25 as reported by the official ROUGE script. The METEOR improvement from the 50k baseline to the pointer-generator model, and from the pointer-generator to the pointer-generator+coverage model, were both found to be statistically significant using an approximate randomization test with $p < 0.01$.

3.6 Results

3.6.1 Preliminaries

Our results are given in Table 3.1. We evaluate our models with the standard ROUGE metric (Lin, 2004b), reporting the F_1 scores for ROUGE-1, ROUGE-2 and ROUGE-L (which respectively measure the word-overlap, bigram-overlap, and longest common sequence between the reference summary and the summary to be evaluated). We obtain our ROUGE scores using the `pyrouge` package.[6] We also evaluate with the METEOR metric (Denkowski and Lavie, 2014), both in exact match mode (rewarding only exact matches between words) and full mode (which additionally rewards matching stems, synonyms and paraphrases).[7]

In addition to our own models, we also report the lead-3 baseline (which uses the first three sentences of the article as a summary), and compare to the only previous abstractive

[6] `pypi.python.org/pypi/pyrouge/0.1.3`
[7] `www.cs.cmu.edu/~alavie/METEOR`

(Nallapati et al., 2016) and extractive (Nallapati et al., 2017) models on the full dataset. The output of our models is available online.[8]

Given that we generate plain-text summaries but Nallapati et al. (2016; 2017) generate anonymized summaries (see Section 3.4), our ROUGE scores are not strictly comparable. There is evidence to suggest that the original-text dataset may result in higher ROUGE scores in general than the anonymized dataset – the lead-3 baseline is higher on the former than the latter. One possible explanation is that multi-word named entities lead to a higher rate of n-gram overlap. Unfortunately, ROUGE is the only available means of comparison with Nallapati et al.'s work. Nevertheless, given that the disparity in the lead-3 scores is (+1.1 ROUGE-1, +2.0 ROUGE-2, +1.1 ROUGE-L) points respectively, and our best model scores exceed Nallapati et al. (2016) by (+4.07 ROUGE-1, +3.98 ROUGE-2, +3.73 ROUGE-L) points, we may estimate that we outperform the only previous abstractive system by at least 2 ROUGE points all-round.

3.6.2 Observations

We find that both our baseline models perform poorly with respect to ROUGE and METEOR, and in fact the larger vocabulary size (150k) does not seem to help. Even the better-performing baseline (with 50k vocabulary) produces summaries with several common problems. Factual details are frequently reproduced incorrectly, often replacing an uncommon (but in-vocabulary) word with a more-common alternative. For example in Figure 3.1, the baseline model appears to struggle with the rare word *thwart*, producing *destabilize* instead, which leads to the fabricated phrase *destabilize nigeria's economy*. Even more catastrophically, the summaries sometimes devolve into repetitive nonsense, such as the third sentence produced by the baseline model in Figure 3.1. In addition, the baseline model can't reproduce out-of-vocabulary words (such as *muhammadu buhari* in Figure 3.1). Further examples of all these problems are provided in Section 3.8.

Our pointer-generator model achieves much better ROUGE and METEOR scores than the baseline, despite many fewer training epochs. The difference in the summaries is also marked: out-of-vocabulary words are handled easily, factual details are almost always copied

[8] `www.github.com/abisee/pointer-generator`

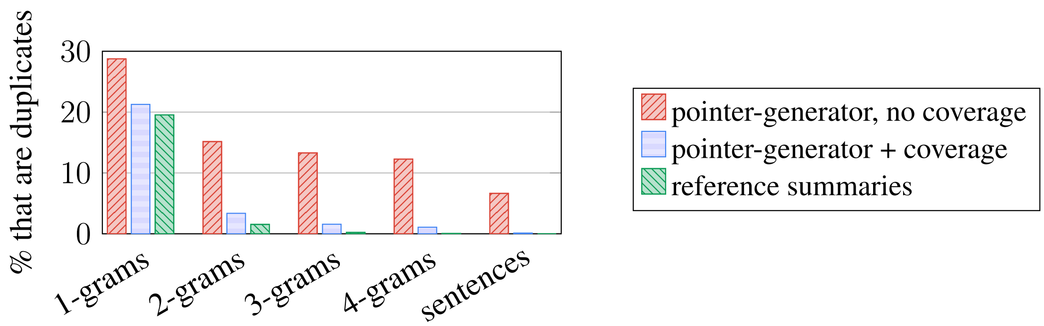


Figure 3.4: Coverage eliminates undesirable repetition. Summaries from our **non-coverage model** contain many duplicated n-grams while our **coverage model** produces a similar number as the **reference summaries**.

correctly, and there are no fabrications (see Figure 3.1). However, repetition is still very common.

Our pointer-generator model with coverage improves the ROUGE and METEOR scores further, convincingly surpassing the best abstractive model of Nallapati et al. (2016) by several ROUGE points. Despite the brevity of the coverage training phase (about 1% of the total training time), the repetition problem is almost completely eliminated, which can be seen both qualitatively (Figure 3.1) and quantitatively (Figure 3.4). However, our best model does not quite surpass the ROUGE scores of the lead-3 baseline, nor the best extractive model at the time of our work (Nallapati et al., 2017). We discuss this issue in section 3.7.1.

3.7 Discussion

3.7.1 Comparison with Extractive Systems

It is clear from Table 3.1 that extractive systems tend to achieve higher ROUGE scores than abstractive, and that the extractive lead-3 baseline is extremely strong (even the best extractive system beats it by only a small margin). We offer two possible explanations for these observations.

Firstly, news articles tend to be structured with the most important information at the start (Pöttker, 2003); this partially explains the strength of the lead-3 baseline. Indeed, we found that using only the first 400 tokens (about 20 sentences) of the article yielded

Article: smugglers lure arab and african migrants by offering discounts to get onto overcrowded ships if people bring more potential passengers, a cnn investigation has revealed. (...)
Summary: cnn investigation **uncovers** the **business inside** a **human smuggling ring**.

Article: eyewitness video showing white north charleston police officer michael slager shooting to death an unarmed black man has exposed discrepancies in the reports of the first officers on the scene. (...)
Summary: more **questions than answers emerge** in **controversial s.c.** police shooting.

Figure 3.5: Examples of highly abstractive human-written reference summaries (**bold** denotes novel words).

significantly higher ROUGE scores than using the first 800 tokens.

Secondly, the nature of the task and the ROUGE metric make extractive approaches and the lead-3 baseline difficult to beat. The choice of content for the reference summaries is quite subjective – sometimes the sentences form a self-contained summary; other times they simply showcase a few interesting details from the article. Given that the articles contain 39 sentences on average, there are many equally valid ways to choose 3 or 4 highlights in this style. Abstraction introduces even more options (choice of phrasing), further decreasing the likelihood of matching the reference summary. For example, *smugglers profit from desperate migrants* is a valid alternative abstractive summary for the first example in Figure 3.5, but it scores 0 ROUGE with respect to the reference summary. This inflexibility of ROUGE is exacerbated by only having one reference summary, which has been shown to lower ROUGE's reliability compared to multiple reference summaries (Lin, 2004a).

Due to the subjectivity of the task and thus the diversity of valid summaries, it seems that ROUGE rewards safe strategies such as selecting the first-appearing content, or preserving original phrasing. While the reference summaries *do* sometimes deviate from these techniques, those deviations are unpredictable enough that the safer strategy obtains higher ROUGE scores on average. This may explain why extractive systems tend to obtain higher ROUGE scores than abstractive, and even extractive systems do not significantly exceed the lead-3 baseline.

To explore this issue further, we evaluated our systems with the METEOR metric, which rewards not only exact word matches, but also matching stems, synonyms and paraphrases (from a pre-defined list). We observe that all our models receive over 1 METEOR point

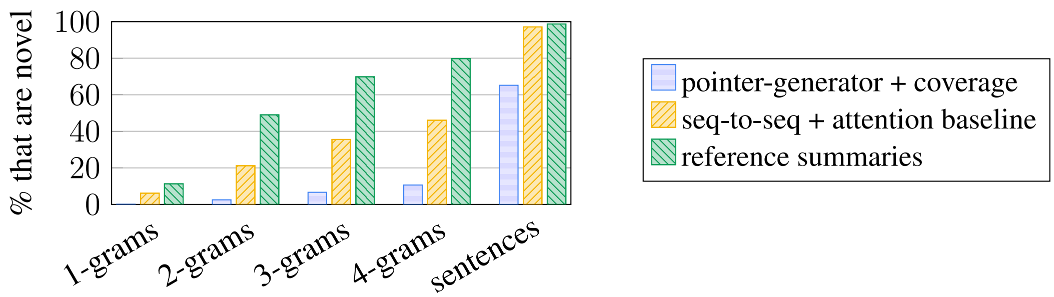


Figure 3.6: Although our **best model** is abstractive, it does not produce novel n-grams (i.e., n-grams that don't appear in the source text) as often as the **reference summaries**. The **baseline model** produces more novel n-grams, but many of these are erroneous (see section 3.7.2).

boost by the inclusion of stem, synonym and paraphrase matching, indicating that they may be performing some abstraction. However, we again observe that the lead-3 baseline is not surpassed by our models. It may be that news article style makes the lead-3 baseline very strong with respect to any metric. We believe that investigating this issue further is an important direction for future work.

3.7.2 How Abstractive Is Our Model?

We have shown that our pointer mechanism makes our abstractive system more reliable, copying factual details correctly more often. But does the ease of copying make our system any less *abstractive*?

Figure 3.6 shows that our final model's summaries contain a much lower rate of novel n-grams (i.e., those that don't appear in the article) than the reference summaries, indicating a lower degree of abstraction. Note that the baseline model produces novel n-grams more frequently – however, this statistic includes all the incorrectly copied words, *UNK* tokens and fabrications alongside the good instances of abstraction.

In particular, Figure 3.6 shows that our final model copies whole article sentences 35% of the time; by comparison the reference summaries do so only 1.3% of the time. This is a main area for improvement, as we would like our model to move beyond simple sentence extraction. However, we observe that the other 65% encompasses a range of abstractive techniques. Article sentences are truncated to form grammatically-correct shorter versions,

Article: andy murray (...) is into the semi-finals of the miami open , but not before getting a scare from 21 year-old austrian dominic thiem, who pushed him to 4-4 in the second set before going down 3-6 6-4, 6-1 in an hour and three quarters. (...)
Summary: andy murray **defeated** dominic thiem 3-6 6-4, 6-1 in an hour and three quarters.

Article: (...) wayne rooney smashes home during manchester united 's 3-1 win over aston villa on saturday. (...)
Summary: manchester united **beat** aston villa 3-1 at old trafford on saturday.

Figure 3.7: Examples of abstractive summaries produced by our model (**bold** denotes novel words).

and new sentences are composed by stitching together fragments. Unnecessary interjections, clauses and parenthesized phrases are sometimes omitted from copied passages. Some of these abilities are demonstrated in Figure 3.1, and Section 3.8 contains more examples.

Figure 3.7 shows two examples of more impressive abstraction – both with similar structure. The dataset contains many sports stories whose summaries follow the *X beat Y* $\langle$*score*$\rangle$ *on* $\langle$*day*$\rangle$ template, which may explain why our model is most confidently abstractive on these examples. In general however, our model does not routinely produce summaries like those in Figure 3.7, and is not close to producing summaries like in Figure 3.5.

The value of the generation probability p_{gen} also gives a measure of the abstractiveness of our model. During training, p_{gen} starts with a value of about 0.30 then increases, converging to about 0.53 by the end of training. This indicates that the model first learns to mostly copy, then learns to generate about half the time. However at test time, p_{gen} is heavily skewed towards copying, with a mean value of 0.17. The disparity is likely due to the fact that during training, the model receives word-by-word supervision in the form of the reference summary, but at test time it does not. Nonetheless, the generator module is useful even when the model is copying. We find that p_{gen} is highest at times of uncertainty such as the beginning of sentences, the join between stitched-together fragments, and when producing periods that truncate a copied sentence. Our mixture model allows the network to copy while simultaneously consulting the language model – enabling operations like stitching and truncation to be performed with grammaticality. In any case, encouraging the pointer-generator model to write more abstractively, while retaining the accuracy advantages of the pointer module, is an exciting direction for future work.

3.8 Examples

This section provides examples from the test set, with side-by-side comparisons of the reference summaries and the summaries produced by our models. In each example:

- *italics* denote out-of-vocabulary words
- **red** denotes factual errors in the summaries
- green shading intensity represents the value of the generation probability p_{gen}
- yellow shading intensity represents final value of the coverage vector at the end of final model's summarization process.

Article (truncated): andy murray came close to giving himself some extra preparation time for his wedding next week before ensuring that he still has unfinished tennis business to attend to . the world no 4 is into the semi-finals of the miami open , but not before getting a scare from 21 year-old austrian dominic *thiem* , who pushed him to 4-4 in the second set before going down 3-6 6-4 , 6-1 in an hour and three quarters . murray was awaiting the winner from the last eight match between tomas berdych and argentina 's juan monaco . prior to this tournament *thiem* lost in the second round of a challenger event to soon-to-be new brit *aljaz* bedene . andy murray pumps his first after defeating dominic *thiem* to reach the miami open semi finals . *muray* throws his *sweatband* into the crowd after completing a 3-6 , 6-4 , 6-1 victory in florida . murray shakes hands with *thiem* who he described as a ' strong guy ' after the game . and murray has a fairly simple message for any of his fellow british tennis players who might be agitated about his imminent arrival into the home ranks : do n't complain . instead the british no 1 believes his colleagues should use the assimilation of the world number 83 , originally from slovenia , as motivation to better themselves .

Reference Summary:
british no 1 defeated dominic *thiem* in miami open quarter finals .
andy murray celebrated his 500th career win in the previous round .
third seed will play the winner of tomas berdych and juan monaco in the semi finals of the atp masters 1000 event in key biscayne .

Baseline:
andy murray beat UNK **bedene 6-3** , 6-4 , 6-1 in an hour and three quarters .
british no 1 believes his colleagues should use the **maze** of the world number 83 , originally from slovenia , as motivation to better themselves .

Pointer-Generator, No Coverage:
andy murray came close to giving himself some extra preparation for his wedding next week before getting a scare from 21 year-old austrian dominic *thiem* .
the world no 4 is into the semi-finals of the miami open , but not before getting a scare from 21 year-old austrian dominic *thiem* .

Pointer-Generator, With Coverage:
andy murray defeated dominic *thiem* 3-6 6-4 , 6-1 in an hour and three quarters .
murray was awaiting the winner from the last eight match between tomas berdych and argentina 's juan monaco .
prior to this tournament thiem lost in the second round of a challenger event to soon-to-be new brit *aljaz* bedene .

Figure 3.8: The baseline model reports the wrong score *6-3*, substitutes *bedene* for *thiem* and struggles with the uncommon word *assimilation*. The pointer-network models accurately reproduce the out-of-vocabulary words *thiem* and *aljaz*. Note that the final model produces the novel word *defeated* to incorporate several fragments into a single sentence.

Article (truncated): munster have signed new zealand international francis *saili* on a two-year deal . utility back *saili* , who made his all blacks debut against argentina in 2013 , will move to the province later this year after the completion of his 2015 contractual commitments . the 24-year-old currently plays for *auckland-based* super rugby side the blues and was part of the new zealand under-20 side that won the junior world championship in italy in 2011 . *saili* 's signature is something of a coup for munster and head coach anthony foley believes he will be a great addition to their backline . francis *saili* has signed a two-year deal to join munster and will link up with them later this year . ' we are really pleased that francis has committed his future to the province , ' foley told munster 's official website . ' he is a talented centre with an impressive *skill-set* and he possesses the physical attributes to excel in the northern hemisphere . ' i believe he will be a great addition to our backline and we look forward to welcoming him to munster . ' *saili* has been capped twice by new zealand and was part of the under 20 side that won the junior championship in 2011 .

Reference Summary:
utility back francis *saili* will join up with munster later this year .
the new zealand international has signed a two-year contract .
saili made his debut for the all blacks against argentina in 2013 .

Baseline:
dutch international francis UNK has signed a two-year deal to join **irish** UNK super rugby side the blues .
UNK 's signature is something of a coup for munster and his head coach anthony foley believes he will be a great addition to their **respective prospects** .
UNK has been capped twice by new zealand .

Pointer-Generator, No Coverage:
new zealand international francis *saili* will move to the province later this year .
utility back *saili* made his all blacks debut against argentina in 2013 .
utility back *saili* will move to the province later this year .

Pointer-Generator, With Coverage:
francis *saili* has signed a two-year deal to join munster later this year .
the 24-year-old was part of the new zealand under-20 side that won the junior world championship in italy in 2011 .
saili 's signature is something of a coup for munster and head coach anthony foley .

Figure 3.9: The baseline model incorrectly substitutes *dutch* for *new zealand* (perhaps reflecting the dataset's European bias), fabricates *irish*, and struggles with out-of-vocabulary words *saili* and *auckland-based*. Though it is not clear why, the phrase *addition to our backline* is changed to the nonsensical *addition to their respective prospects*. The pointer-generator model fixes these accuracy problems, and adding coverage fixes the repetition. Note that the final model skips over large passages of text to produce shorter sentences.

Article (truncated): right from the moment he breezed through the doors at old trafford , louis van gaal was a man with a plan . the first season , he stated , would see him deliver manchester united back into their rightful place in the champions league . he would restore them to the premier league top four but *loftier* aims of silverware would have to wait . his three-year vision would allow for such thoughts but , first things first , united needed to be dining from european football 's top table again . louis van gaal is close to delivering his *first-season* aim of returning man united into champions league . wayne rooney smashes home during manchester united 's 3-1 win over aston villa on saturday . united 's win over aston villa took them third , eight points ahead of fifth-placed liverpool in the table . april 12 manchester city (h) . april 18 chelsea (a) . april 26 everton (a) . may 2 west bromwich albion (h) . may 9 crystal palace (a) . may 17 arsenal (h) . may 24 hull city (a) . one season out of the champions league was far from ideal , but two seasons would be an absolute disaster and something , he understood , that would not be tolerated . in november , even that was looking optimistic . a 1-0 defeat to manchester city meant that united had taken just 13 points from their opening 10 matches - it was their worst start to a league campaign since 1986 , when ron atkinson only lasted until november .

Reference Summary:
man united have an eight-point cushion from *fifth-place* liverpool .
van gaal looks likely to deliver on his promise of top four finish .
but the dutchman has a three-year vision mapped out .
next season will have to see united mount sustained challenge for title .
they must also reach the later stages of the champions league .

Baseline:
manchester united beat aston villa 3-1 at old trafford on saturday .
louis van gaal is close to delivering his UNK aim of returning man united into the premier league top four .
louis van gaal is close to delivering his UNK aim of returning man united into champions league .

Pointer-Generator, No Coverage:
louis van gaal is close to delivering his *first-season* aim of returning man united into champions league.
united 's win over aston villa took them third , eight points ahead of fifth-placed liverpool in the table .
louis van gaal is close to delivering his *first-season* aim of returning man united into champions league.

Pointer-Generator, With Coverage:
manchester united beat aston villa 3-1 at old trafford on saturday .
louis van gaal is close to delivering his *first-season* aim of returning man united into champions league .
united needed to be dining from european football 's top table again .

Figure 3.10: In this example, both our baseline model and final model produce a completely abstractive first sentence, using a novel word *beat*.

Article (truncated): having been on the receiving end of a 6-1 thumping , a defeat like that could be justifiably met with a backlash by angry supporters . watching a 3-1 first leg aggregate advantage turn into a 7-4 deficit come the end of the reverse encounter too could send many fans *apoplectic* at the capitulation of their side . however that does n't appear the case for those devoted to porto . porto supporters gave their team a hero 's welcome following their 6-1 defeat at bayern munich on tuesday . porto star striker jackson martinez was one of many players to look perplexed by their warm reception . porto boss *julen* *lopetegui* (left) was hugged by fans congratulating him on their champions league run . police escorts were needed to keep the delirious supporters at bay as the porto team bus drove past . the team bus was met with a cacophony of noise from porto supporters proudly chanting about their club . on their return from a humiliating champions league quarter-final loss at the hands of bayern munich on tuesday night , the squad were given a heroes reception as they arrived back in portugal . [...] it was the first time that porto , who had been unbeaten in this season 's tournament up until tuesday night , had reached the quarter-finals of the champions league since the 2008-09 season .

Reference Summary:
bayern munich beat porto 6-1 in their champions league tie on tuesday .
result saw bayern win quarter-final encounter 7-4 on aggregate .
it was the first-time porto had reached that stage since the 2008-09 season .

Baseline:
porto **beat** bayern munich **2-0** in the champions league on tuesday night .
porto star **james** UNK was one of many players involved in the match .
the squad were given a **trophy** as they arrived back in portugal .

Pointer-Generator, No Coverage:
porto star striker jackson martinez was one of many players to look perplexed by their warm reception.
porto star striker jackson martinez was one of many players to look perplexed by their warm reception.
porto star striker jackson martinez was one of many players to look perplexed by their warm reception.

Pointer-Generator, With Coverage:
porto star striker jackson martinez was one of many players to look perplexed by their warm reception .
porto boss julen lopetegui was hugged by fans congratulating him on their champions league run .
however that does n't appear the case for those devoted to porto .
police escorts were needed to keep the delirious supporters at bay .

Figure 3.11: The baseline makes several inaccuracies: it claims *porto* beat *bayern munich* not vice versa, the score is changed from *7-4* to *2-0*, *jackson* is changed to *james* and *a heroes reception* is replaced with *a trophy*. Our final model produces sentences that are individually accurate, but nonsensical as a whole. Note that the final model omits the parenthesized phrase *(left)* from its second sentence.

Article: (cnn) " it 's showtime ! " michael keaton paid homage – ever so slightly – to his roles in " *beetlejuice* " and " batman " in his third turn hosting " saturday night live " this weekend . keaton acknowledged in his opening monologue that a lot has changed since he first hosted the comedy sketch show in 1982 . " i had a baby – he 's 31 . i also have a new girlfriend – she 's 28 , " he said . fans who were hoping for a full-blown revival of keaton 's most memorable characters might have been a little disappointed . snl cast members *taran killam* and bobby moynihan begged the actor with a song to " play " batman and *beetlejuice* with them . all they got in response were a couple of one-liners . overall , keaton 's performance drew high marks from viewers and critics for its " deadpan " manner and " unpredictable *weirdness* , " in the words of *a.v* . club 's dennis perkins . fans also delighted in a cameo from " walking dead " star norman *reedus* during weekend update . keaton scored some laughs from the audience as an ad executive who 's not very good at his job , a confused grandfather and a high school teacher who gets asked to the prom in a riff on the romantic comedy " she 's all that . " other *crowd-pleasing* spots included a scientology parody music video and a news conference *spoofing* the ncaa *student-athlete* debate . the show also poked fun at cnn with cast member cecily strong playing anchor brooke baldwin . baldwin said on twitter that she 's taking it as a " crazy compliment " and shared a clip from the skit .

Reference Summary:
michael keaton hosted " saturday night live " for the first time in 1982 .
in 2015 , his nods to starring roles in " *beetlejuice* " and " batman " are brief .

Baseline:
" snl " wins " saturday night live " .
keaton acknowledged in his opening " saturday night live " .
keaton acknowledged in his opening night 's " UNK " .
the show also pokes fun at cnn with cast member **mariah carey** .

Pointer-Generator, No Coverage:
" *beetlejuice* " michael keaton paid homage to his roles in " *beetlejuice* " and " batman " in his third turn hosting " saturday night live " this weekend .
keaton acknowledged in his opening monologue that a lot has changed since he first hosted the comedy sketch show in 1982 .

Pointer-Generator, With Coverage:
michael keaton paid homage to his roles in " *beetlejuice* " and " batman " in his third turn hosting " saturday night live " .
keaton acknowledged in his opening monologue that a lot has changed since he first hosted the comedy sketch show in 1982 .

Figure 3.12: Baseline model is generally nonsensical, and replaces *cecily strong* with *mariah carey*; it may be struggling with the out-of-vocabulary *beetlejuice*, or the unusual non-news article format. Note the final model omits – *ever so slightly* – from its first sentence.

Article (truncated): they are supposed to be the dream team who can solve the conundrum of how to win the election . but david cameron and boris johnson were left scratching their heads today as they struggled with a children 's jigsaw teaching toddlers about the seasons . as the london mayor tried to hammer ill-fitting pieces together with his hands , the prime minister tried out what could be a new election slogan , telling him : ' if in doubt , wedge it in . ' after being put right by a four-year-old who spotted their errors , the pair had more fun finger painting with tory blue paint . david cameron and boris johnson were left stumped by the puzzle at advantage children 's day nursery in *surbiton* , as three-year-old stephanie looked on . when they tried to put the puzzle back together , they hit trouble after it proved to be more difficult than expected . [...] they were supposed to be highlighting tory plans to double free childcare for 600,000 children aged three and four . they sat at a table with joshua and stephanie , both aged three , and four-year-olds hamish and leo where they broke up a puzzle which depicted the four seasons . when they tried to put it back together , they hit trouble after it proved to be more difficult than expected .

Reference Summary:
david cameron and boris johnson visit nursery to promote childcare .
duo left *flummoxed* by jigsaw aimed at teaching children about seasons .
london mayor resorted to brute force to hammer wrong pieces together .
prime minister gave his approval , telling boris : ' if in doubt , wedge it in ' .

Baseline:
david cameron and boris johnson were left mystified by the puzzle .
they were at the nursery in UNK to promote a tory pledge to double free childcare for **25** children aged three and four .
the pair were at the nursery in UNK to promote a tory pledge to double free childcare for children aged three and four .

Pointer-Generator, No Coverage:
london mayor tried to hammer ill-fitting pieces together with his hands .
they were supposed to be highlighting tory plans to double free childcare .
they were supposed to be highlighting tory plans to double free childcare .
they were supposed to be highlighting tory plans to double free childcare .

Pointer-Generator, With Coverage:
london mayor tried to hammer ill-fitting pieces together with his hands .
prime minister tried out what could be a new election slogan .
but they hit trouble after it proved to be more difficult than expected .

Figure 3.13: The baseline appropriately replaces *stumped* with novel word *mystified*; however this is not rewarded by ROUGE as the reference summary instead chooses *flummoxed*. The baseline incorrectly substitutes *600,000* for *25*. In the final model's output, generation probability is largest at the beginning of sentences (especially the first verb) and on periods.

Article (truncated): lagos , nigeria (cnn) a day after winning nigeria 's presidency , *muhammadu buhari* told cnn 's christiane amanpour that he plans to aggressively fight corruption that has long plagued nigeria and go after the root of the nation 's unrest . *buhari* said he 'll " rapidly give attention " to curbing violence in the northeast part of nigeria , where the terrorist group boko haram operates . by cooperating with neighboring nations chad , cameroon and niger , he said his administration is confident it will be able to thwart criminals and others contributing to nigeria 's instability . for the first time in nigeria 's history , the opposition defeated the ruling party in democratic elections . *buhari* defeated incumbent goodluck jonathan by about 2 million votes , according to nigeria 's independent national electoral commission . the win comes after a long history of military rule , coups and botched attempts at democracy in africa 's most populous nation .

Reference Summary:
muhammadu buhari tells cnn 's christiane amanpour that he will fight corruption in nigeria .
nigeria is the most populous country in africa and is grappling with violent boko haram extremists .
nigeria is also africa 's biggest economy , but up to 70 % of nigerians live on less than a dollar a day .

Baseline:
UNK UNK says his administration is confident it will be able to **destabilize nigeria 's economy** .
UNK says his administration is confident it will be able to thwart criminals and other **nigerians** .
he says the country has long nigeria and nigeria 's economy .

Pointer-Generator, No Coverage:
muhammadu buhari says he plans to aggressively fight corruption in the northeast part of nigeria .
he says he 'll " rapidly give attention " to curbing violence in the northeast part of nigeria .
he says his administration is confident it will be able to thwart criminals .

Pointer-Generator, With Coverage:
muhammadu buhari says he plans to aggressively fight corruption that has long plagued nigeria .
he says his administration is confident it will be able to thwart criminals .
the win comes after a long history of military rule , coups and botched attempts at democracy in africa 's most populous nation .

Figure 3.14: The baseline model incorrectly changes *thwart criminals and others contributing to nigeria's instability* to *destabilize nigeria's economy* – which has a mostly opposite meaning. It also produces a nonsensical sentence. Note that our final model produces the novel word *says* to paraphrase *told cnn 's christiane amanpour*.

Article (truncated): cairo (cnn) at least 12 people were killed sunday , and more injured , in separate attacks on a police station , a checkpoint and along a highway in egypt 's northern sinai , authorities said . six people , including one civilian , were killed when a car bomb exploded near the police station in *al-arish* , capital of north sinai , health ministry spokesman *hossam abdel-ghafar* told ahram online . he said 40 people were injured . ansar beit *al-maqdis* , an isis affiliate , claimed responsibility for the attack , which came hours after another operation that the group also claimed . in that earlier attack , a first lieutenant , a sergeant and four *conscripts* were killed when their armored vehicle was attacked on the highway from *al-arish* to sheikh *zuweid* in northern sinai , the military said . two other soldiers were injured and taken to a military hospital . ansar beit *al-maqdis* has claimed many attacks against the army and police in sinai . a third attack sunday on a checkpoint in rafah left three security personnel injured , after unknown assailants opened fire at them , according to state media . the attacks come as the military announced a reshuffle of several senior military positions , state media reported . among those being replaced are the generals in charge of military intelligence and egypt 's second field army , which is spearheading the battle against the insurgents in the northern sinai . egypt 's army has been fighting a decade-long militant islamist insurgency , which has spiked since the ouster of muslim brotherhood president mohamed morsy in the summer of 2013 . hundreds of police and soldiers , as well as civilians , have been killed in militant attacks in the past months .

Reference Summary:
six people , including one civilian , are killed when a car bomb explodes near a police station .
six others are killed when their armored vehicle is attacked on a highway in northern sinai .
ansar beit *al-maqdis* , an isis affiliate , claims responsibility .

Baseline:
he says the attacks come after another operation that killed 12 .
the attacks come as military announced a **u.n. peacekeeping force** in northern sinai .

Pointer-Generator, No Coverage:
ansar beit *al-maqdis* , an isis affiliate , claimed responsibility for the attack .
ansar beit *al-maqdis* , an isis affiliate , claimed responsibility for the attack .
the attacks come as the military announced a reshuffle of several senior military positions .

Pointer-Generator, With Coverage:
six people , including one civilian , were killed when a car bomb explodes near the police station .
ansar beit *al-maqdis* , an isis affiliate , claimed responsibility for the attack .
egypt 's army has been fighting a decade-long militant islamist insurgency .

Figure 3.15: The baseline fabricates *a u.n. peacekeeping force* unmentioned in the article – perhaps inspired by a connection between U.N. peacekeeping forces and *northern sinai* in the training data. The pointer-generator model is more accurate, correctly reporting the *reshuffle of several senior military positions*.

3.9 Conclusion

In this chapter we presented a hybrid pointer-generator architecture with coverage, and showed that it reduces inaccuracies and repetition. We applied our model to a challenging long-text dataset, and significantly outperformed the previous abstractive state-of-the-art ROUGE scores. Our model is capable of exhibiting many abstractive abilities, but its outputs are still a lot less abstractive than the human-written summaries.

Since the publication of this work, there have been several developments in neural generative abstractive summarization, some of which I summarize here:

Datasets. Kryściński et al. (2019) find that the inverted pyramid structure of news-based datasets like *CNN/Daily Mail* biases the learning signal too strongly, resulting in a loss of diversity. Furthermore, web-scraped datasets like *CNN/Daily Mail* contain noise such as HTML code, hyperlinks and clickbait (Kryściński et al., 2019; Fabbri et al., 2021). Several higher-quality summarization datasets have been introduced for various genres including news (Narayan et al., 2018), scientific and medical publications (Cohan et al., 2018), patents (Sharma et al., 2019) and congressional bills (Kornilova and Eidelman, 2019).

Evaluation. The community has become increasingly aware that ROUGE is a poor proxy for summary quality; Kryściński et al. (2019) show that for model-generated abstractive summaries on the *CNN/Daily Mail* dataset, ROUGE has minimal correlation with human judgments of relevance, consistency, fluency, and coherence. While there are many alternative automatic metrics, in a large-scale evaluation Fabbri et al. (2021) find that in general, they have poor correlation with coherence and relevance, and a better correlation with consistency and fluency – at least for the summaries generated by currently-available summarization systems. In general, the research community has moved increasingly towards human evaluation.

Models. The best results have generally been achieved by larger pretrained Transformers. For example, T5 (Raffel et al., 2020) provides a unified framework for many text-to-text tasks including abstractive summarization, while PEGASUS (Zhang et al., 2020a) specifically targets abstractive summarization with a tailored pretraining objective. Over several datasets, the largest PEGASUS model produces summaries that are judged as high-quality as the human-written reference summaries, when judged by crowdworkers. However, the summaries produced by PEGASUS are still less abstractive than the references. Greedy decoding and beam search remain the most commonly-used decoding algorithms for abstractive summarization. While copying and coverage mechanisms similar to those used in this chapter appeared frequently in RNN-based approaches, they are largely absent from Transformer-based approaches Syed et al. (2021).

Factual accuracy. Through a manual annotation of summaries produced by recent abstractive summarization models, Kryściński et al. (2019) find that factual inaccuracies are common (30%); Cao et al. (2018) found similar results. Factual accuracy has become a focus of summarization research, with many proposed methods to evaluate (Kryscinski et al., 2020; Pagnoni et al., 2021) and improve (Cao et al., 2018; Zhu et al., 2021; Zhang et al., 2020c) factual accuracy.

Chapter 4

Controlling Attributes of Chitchat Dialogue

4.1 Introduction

We now turn to chitchat dialogue, a significantly more open-ended task than abstractive summarization (Figure 1.1). Though utterances generated in this chapter are much shorter than the summaries in the previous chapter, the dialogue setting adds the complexity of multi-turn interaction with users. In this more complex setting we now must consider the effects of the bot's actions over multiple turns, and the actions of the user.

Neural generation models for dialogue, which became ubiquitous in research of the late 2010s, are still poorly understood. Well known problems, such as the genericness and repetitiveness of responses (Serban et al., 2016b), remain without a *de facto* solution. Strikingly, the factors that determine human judgments of overall conversation quality are almost entirely unexplored. Most works have been limited to the next utterance prediction problem, whereas a multi-turn evaluation is necessary to evaluate the quality of a full conversation.

In this chapter we both (i) conduct a large-scale study to identify the fine-grained factors governing human judgments of full conversations, and (ii) develop models that apply our findings in practice, leading to state-of-the-art performance. Specifically, we identify and study eight aspects of conversation that can be measured by human judgments, while varying

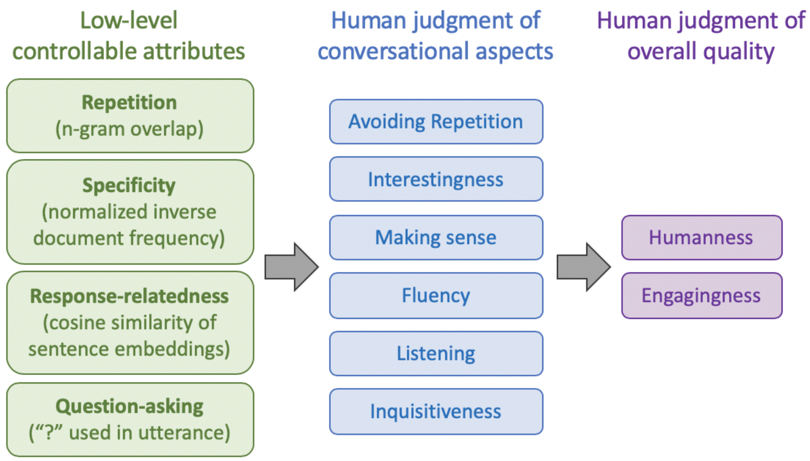


Figure 4.1: We manipulate four low-level attributes and measure their effect on human judgments of individual conversational aspects, as well as overall quality.

four types of low-level attributes that can be algorithmically controlled in neural models; see Figure 4.1. To control the low-level model attributes, we consider two simple but general algorithms: conditional training, in which the neural model is conditioned on additional control features, and weighted decoding, in which control features are added to the decoding scoring function at test time only.

One major result of our findings is that existing work has ignored the importance of conversational flow, as standard models (i) repeat or contradict previous statements, (ii) fail to balance specificity with genericness, and (iii) fail to balance asking questions with other dialogue acts. Conducting experiments on the PersonaChat task (Zhang et al., 2018b), we obtain significantly higher engagingness scores than the baseline by optimizing control of repetition, specificity and question-asking over multiple turns. Using these findings, our best model matches the performance of the winning entry in the NeurIPS 2018 ConvAI2 competition (Dinan et al., 2019a), which was trained on much more data but had no control (see Section 4.9.1). Our code, pretrained models, and full chatlogs, are available at `https://parl.ai/projects/controllable_dialogue`.

4.2 Related Work

Dialogue Dialogue evaluation is relatively well understood in goal-oriented tasks, where automated approaches can be coded by measuring task completion (Bordes et al., 2017; El Asri et al., 2017; Hastie, 2012; Henderson et al., 2014; Wen et al., 2017). Task success combined with dialogue cost can be linked to human judgments like user satisfaction via the PARADISE framework (Walker et al., 1997).

However in chitchat tasks, which we study in this chapter, automatic metrics and their relation to human ratings are less well-understood. While word-overlap metrics are effective for question-answering and machine translation, for dialogue they have little to no correlation with human judgments (Liu et al., 2016; Novikova et al., 2017) – this is due to the open-ended nature of dialogue. There have been attempts to find better automatic approaches, such as adversarial evaluation (Li et al., 2017b) and learning a scoring model (Lowe et al., 2017), but their value is still unclear.

Nevertheless, a number of studies only use automatic metrics, with no human study at all (Lowe et al., 2015; Parthasarathi and Pineau, 2018; Serban et al., 2016a). Other works do use human evaluations (Dinan et al., 2019b; Li et al., 2016a,b; Venkatesh et al., 2017; Vinyals and Le, 2015; Zhang et al., 2018b), typically reporting just one type of judgment (either quality or appropriateness) via a Likert scale or pairwise comparison. Most of those works only consider single turn evaluations, often with a shortened dialogue history, rather than full multi-turn dialogue.

A more comprehensive evaluation strategy has been studied within the scope of the Alexa prize (Venkatesh et al., 2017; Guo et al., 2018) by combining multiple automatic metrics designed to capture various conversational aspects (engagement, coherence, domain coverage, conversational depth and topical diversity). Though these aspects have some similarity to the aspects studied here, we also focus on lower-level aspects (e.g., avoiding repetition, fluency), to understand how they correspond to both our controllable attributes, and to overall quality judgments.

Controllable neural text generation Researchers have proposed several approaches to control aspects of RNN-based natural language generation such as sentiment, length, speaker style and tense (Fan et al., 2018a; Ficler and Goldberg, 2017; Ghazvininejad et al., 2017; Hu et al., 2017; Kikuchi et al., 2016; Peng et al., 2018; Wang et al., 2017). In particular, several works use control to tackle the same common sequence-to-sequence problems we address here (particularly genericness and unrelated output), in the context of single-turn response generation (Baheti et al., 2018; Li et al., 2016a, 2017a; Shen et al., 2017; Xing et al., 2017; Zhang et al., 2018a; Zhou et al., 2017). By contrast, we focus on developing controls for, and human evaluation of, *multi*-turn interactive dialogue – this includes a new method (described in Section 4.5) to control attributes at the *dialogue* level rather than the utterance level.

In this chapter, we require a control method that is both general-purpose (one technique to simultaneously control many attributes) and easily tunable (the control setting is adjustable after training). Given these constraints, we study two control methods: conditional training (variants of which have been described by Fan et al. (2018a); Kikuchi et al. (2016); Peng et al. (2018)) and weighted decoding (described by Ghazvininejad et al. (2017) as a general technique, and by Baheti et al. (2018) to control response-relatedness). To our knowledge, this work was the first to systematically compare the effectiveness of two general-purpose control methods across several attributes.

4.3 The PersonaChat dataset

PersonaChat (Zhang et al., 2018b) is a chitchat dialogue task involving two participants (two humans or a human and a bot). Each participant is given a *persona* – a short collection of personal traits such as *I'm left handed* or *My favorite season is spring* – and are instructed to get to know each other by chatting naturally using their designated personas, for 6–8 turns. The training set contains 8939 conversations and 955 personas, collected via crowdworkers, plus 1000 conversations and 100 personas for validation, and a similar number in the hidden test set. The PersonaChat task was the subject of the NeurIPS 2018 ConvAI2 Challenge (Dinan et al., 2019a), in which competitors were first evaluated with respect to automatic metrics (perplexity, hits@1 and F1 score), and then with respect to human judgment via the question *"How much did you enjoy talking to this user?"* on a scale of 1–4.

4.4 Baseline Model

Our baseline model is a 2-layer LSTM sequence-to-sequence model with attention; see Section 2.3. On any dialogue turn, the input x to the encoder is the entire dialogue history (separated using unique speaker-identifying tokens), with the model's own persona prepended. Conditioned on this input sequence x, the decoder generates a response y. Except when stated otherwise, all our models decode using beam search with beam size 20.

We initialized the word embedding matrix with 300-dimensional GloVe embeddings (Pennington et al., 2014). Using the ParlAI framework (Miller et al., 2017), we pretrained the model on a dataset of 2.5 million Twitter message-response pairs,[1] then fine-tuned it on PersonaChat. On the PersonaChat validation set, the baseline model has a perplexity of 26.83 and F1 of 17.02, which would have placed us 4th out of 26 models in the ConvAI2 competition (Dinan et al., 2019a). We attempt to improve over this baseline using control.

4.5 Controllable Text Generation Methods

Suppose we have a sequence-to-sequence model which gives:

$$P(y|x) = \prod_t P(y_t|x, y_1, \ldots, y_{t-1}) \tag{4.1}$$

This is the conditional probability of a response y (the model's next utterance) given input x (the context, which in our case includes the model's own persona and the dialogue history).

Contrary to most previous work, which controls *at the sentence level*, we wish to control attributes of the output y *at the dialogue level* – meaning that a single control setting is used for a whole dialogue. For example, to control question-asking, we provide a control setting at the beginning of each dialogue (e.g., *20% questions* or *70% questions*) rather than providing a control setting for each utterance (e.g., *is a question* or *isn't a question*). With this approach, the sequence-to-sequence model is able to choose what value the controlled attribute should take for any particular utterance, but we are able to choose the overall

[1] The Twitter dataset is provided in ParlAI; details can be found here: https://parl.ai/docs/tasks.html

distribution. We find that this approach works well – for example, the sequence-to-sequence model is generally good at detecting when to ask a question. In particular, this is easier than the alternative: developing a separate process to decide, for each utterance, whether to ask a question.

In this section, we describe the two methods – which we call Conditional Training (CT) and Weighted Decoding (WD) – that we use to control attributes of the output y at the dialogue level.

4.5.1 Conditional Training (CT)

Conditional Training (Fan et al., 2018a; Kikuchi et al., 2016; Peng et al., 2018) is a method to learn a sequence-to-sequence model $P(y|x, z)$, where z is a discrete *control variable*. If the control attribute is naturally continuous (for example in this chapter, repetitiveness, specificity and response-relatedness), we use z to represent bucketed ranges. For a binary attribute like question-asking, z represents an overall probability (as explained in Section 4.5).

To train a CT model, we first automatically annotate every (x, y) pair in the training set with the attribute we wish to control (for example, whether y contains a question mark). During training, for each example we determine the corresponding z value (for continuous attributes, this simply means sorting into the correct bucket; for question-asking, see Section 4.6.4). Next, the control variable z is represented via an embedding (each of the possible values of z has its own embedding). For all our experiments, the embedding is of length 10; this was determined via hyperparameter tuning. There are several possible ways to condition the sequence-to-sequence model on z – for example, append z to the end of the input sequence, or use z as the START symbol for the decoder. We find it most effective to concatenate z to the decoder's input on every step.[2] Lastly, the CT model learns to produce $y = y_1, \ldots, y_T$ by optimizing the cross-entropy loss:

$$\text{loss}_{\text{CT}} = -\frac{1}{T}\sum_{t=1}^{T}\log P(y_t|x, z, y_1, \ldots, y_{t-1}) \tag{4.2}$$

[2] To build a CT model $P(y|x, z_1, \ldots, z_n)$ conditioned on *multiple* controls $\{z_1, \ldots, z_n\}$, we can simply concatenate multiple control embeddings to the decoder inputs.

Our CT models are initialized with the parameters from the baseline sequence-to-sequence model $P(y|x)$ (the new decoder parameters are initialized with small random values), then fine-tuned to optimize loss_{CT} on the PersonaChat training set, until convergence of loss_{CT} on the validation set.

4.5.2 Weighted Decoding (WD)

Weighted Decoding (Ghazvininejad et al., 2017) is a decoding method that increases or decreases the probability of words with certain features. The technique is applied only at test time, requiring no change to the training method. A limitation of WD is that the controllable attribute must be defined at the word-level; any desired utterance-level attribute must be redefined via word-level features.

In weighted decoding, on the tth step of decoding, a partial hypothesis $y_{<t} = y_1, \dots, y_{t-1}$ is expanded by computing the score for each possible next word w in the vocabulary:

$$\text{score}(w, y_{<t}; x) = \text{score}(y_{<t}; x) + \log P_{\text{RNN}}(w|y_{<t}, x) + \sum_i w_i f_i(w; y_{<t}, x). \tag{4.3}$$

Here, $\log P_{\text{RNN}}(w|y_{<t}, x)$ is the log-probability of the word w calculated by the RNN, $\text{score}(y_{<t}; x)$ is the accumulated score of the already-generated words in the hypothesis $y_{<t}$, and $f_i(w; y_{<t}, x)$ are *decoding features* with associated weights w_i. There can be multiple features f_i (to control multiple attributes), and the weights w_i are hyperparameters to be chosen.

A decoding feature $f_i(w; y_{<t}, x)$ assigns a real value to the word w, in the context of the text generated so far $y_{<t}$ and the context x. The feature can be continuous (e.g., the unigram probability of w), discrete (e.g., the length of w in characters), or binary (e.g., whether w starts with the same letter as the last word in $y_{<t}$). A positive weight w_i increases the probability of words w that score highly with respect to f_i; a negative weight decreases their probability.

Note that weighted decoding and conditional training can be applied simultaneously (i.e., train a CT model then apply WD at test time) – a strategy we use in our experiments.

Feature	**Condition**
`extrep_bigram`$(w, y_{<t}, x)$	Adding w to the hypothesis $y_{<t}$ would create a 2-gram that appears in a previous utterance by the model
`extrep_unigram`$(w, y_{<t}, x)$	w is a non-stopword **and** w appears in a previous utterance by the model
`intrep_bigram`$(w, y_{<t}, x)$	Adding w to the hypothesis $y_{<t}$ would create a 2-gram that appears earlier in the hypothesis $y_{<t}$
`intrep_unigram`$(w, y_{<t}, x)$	w is a non-stopword **and** w appears earlier in the hypothesis $y_{<t}$
`partnerrep_bigram`$(w, y_{<t}, x)$	Adding w to the hypothesis $y_{<t}$ would create a 2-gram that appears in a previous utterance by the partner

Table 4.1: We define five binary features for controlling different types of repetition via weighted decoding. Each feature depends on the word w, the partial hypothesis $y_{<t}$, and the context x (which includes the model's own persona and the dialogue history). Each of these features is equal to 1 if and only if the condition on the right is true; otherwise 0.

4.6 Controlling Conversational Attributes

In this section, we describe how we use conditional training and weighted decoding to control four attributes: repetition, specificity, response-relatedness and question-asking. We evaluate the effectiveness of both control methods via automatic metrics (i.e., measuring how well the attribute was controlled) – these are summarized in Table 4.6. We later use these findings to select control methods and control settings to be explored further via human evaluation (Section 4.9).

4.6.1 Repetition

Our baseline model exhibits three types of repetition, which we call *external repetition* (self-repetition across utterances), *internal repetition* (self-repetition within utterances), and *partner repetition* (repeating the conversational partner).

To control repetition with weighted decoding,[3] we define five n-gram based decoding features (see Table 4.1). Three of these features (`extrep_bigram`, `intrep_bigram` and

[3] We also tried controlling repetition with conditional training, defining z as the (bucketed) maximum ROUGE-L precision between the response y and the bot's previous utterances. However, this method was unsuccessful because there are not enough repetitive examples in the training data for the model to learn the control. Experimenting with data augmentation to solve this problem is an area for future work.

Input: *Yes, I'm studying law at the moment*
Baseline Response: *That sounds like a lot of fun!*

Weight	NIDF	Weighted Decoding Response
-5.0	0.6%	*Oh......................................*
0.0	17.1%	*That sounds like a lot of fun!*
3.0	18.3%	*That sounds like a lot of fun. How long have you been studying?*
7.0	38.5%	*I majored in practising my spiritual full time philosophy test*
10.0	71.9%	*Oh wow! Merna jean isa paino yi hao hui bu acara sya gila* [. . .]

z	NIDF	Conditional Training Response
0	16.8%	*Sounds like you are a great person!*
2	18.3%	*So you are a law student?*
4	18.4%	*That sounds like a lot of fun*
6	22.8%	*That sounds like a rewarding job!*
8	24.4%	*That sounds like a rewarding career!*

Table 4.2: Middle: Example of controlling specificity (NIDF) via weighted decoding. At the extremes, the model produces only the most rare or the most common tokens. Bottom: Example of controlling specificity via conditional training. This gives a narrower NIDF range, but all the responses are appropriate.

`partnerrep_bigram`) identify repeating bigrams for the three repetition types. The other two features (`extrep_unigram` and `intrep_unigram`) identify repeating content words. By applying a negative weight to these features, we can reduce repetition. In particular, if the weight is $-\infty$, our method is equivalent to *n-gram blocking* as described by Kulikov et al. (2019). We observe that repetition control is very important, thus all further control experiments include repetition control.

4.6.2 Specificity

Like many sequence-to-sequence models using beam search decoding, our baseline frequently asks generic questions such as *What music do you like?* and gives dull, unspecific responses, such as *I like all kinds of music*.

We control specificity using Normalized Inverse Document Frequency (NIDF) as a measure of word rareness.[4] The Inverse Document Frequency of a word w is $\text{IDF}(w) =$

[4]Note that our NIDF specificity features are similar to the NIRF and NIWF features used by Zhang et al. (2018a).

Input: *Do you go get coffee often*
Baseline Response: *I do, when I am not playing the piano.*

Weight	Cos sim	Weighted Decoding Response
-10.0	-0.05	*I am a musician.*
0.0	-0.02	*I do, when I am not playing the piano.*
5.0	0.40	*I do, usually at starbucks.*
8.0	0.59	*Not usually, especially when you drink latte.*
11.0	0.72	*Not often, usually with drinks, espresso, latte, tea, etc.*

Table 4.3: Example of controlling response-relatedness (cosine similarity to input) via weighted decoding. Positive weights (e.g., 5.0) can yield more on-topic responses, but higher weights (e.g., 11.0) can result in nonsensical lists of topically related words.

$\log(R/c_w)$ where R is the number of responses in the dataset, and c_w is the number of those responses that contain w. Normalized IDF (which ranges from 0 to 1) is

$$\text{NIDF}(w) = \frac{\text{IDF}(w) - \texttt{min\_idf}}{\texttt{max\_idf} - \texttt{min\_idf}} \tag{4.4}$$

where `min_idf` and `max_idf` are the minimum and maximum IDFs, taken over all words in the vocabulary. To control specificity with weighted decoding, we use NIDF as a decoding feature. As shown in Table 4.2, this method produces reasonable outputs when the feature weight is within a certain range, but at the extremes the outputs are nonsensical. The boundary for nonsensical output differs from example to example.

To control specificity with conditional training, we define the specificity of an utterance y to be the mean NIDF of the words in y. Thus our control variable z is mean NIDF (discretized into 10 equal-sized buckets). As shown in Table 4.2, this method gives outputs with a narrower NIDF range, but overall produces less nonsensical outputs.

4.6.3 Response-relatedness

In conversation, it's generally desirable to produce a response that is related to the partner's last utterance; for example if the partner says *My grandfather died last month*, it is appropriate to say *I'm so sorry. Were you close to your grandfather?* However, our baseline model frequently responds with unrelated utterances like *Do you have any pets?*

To control response-relatedness with weighted decoding, we use the decoding feature $\texttt{resp\_rel}$:

$$\texttt{resp\_rel}(w; y_{<t}, x) = \texttt{cos\_sim}(\texttt{word\_emb}(w), \texttt{sent\_emb}(\ell)) \tag{4.5}$$

where $\texttt{word\_emb}(w)$ is the GloVe embedding for the word w, $\texttt{sent\_emb}(\ell)$ is the sentence embedding for the partner's last utterance ℓ (note ℓ is part of the context x), and $\texttt{cos\_sim}$ is the cosine similarity between the two. In particular, the sentence embedding $\texttt{sent\_emb}(s)$ for an utterance s is a weighted average of the GloVe embeddings of the words in s, with the first principal component projected out; for full details, see Arora et al. (2017). This method of controlling response-relatedness is similar to that described in Baheti et al. (2018). We find that weighted decoding is effective to control the semantic relatedness of the model's response to the partner's last utterance (see Table 4.3). As before, we find that extreme weights lead to nonsensical output.

To control response-relatedness with conditional training, we try defining the control variable z to be $\texttt{cos\_sim}(\texttt{sent\_emb}(y), \texttt{sent\_emb}(\ell))$, the overall cosine similarity between the partner's last utterance ℓ and the model's response y (again, we discretize z). However, we find this method ineffective – the CT model learns only a very weak connection between z and the semantic relatedness of the output (see Section 4.7 for more details).

4.6.4 Question-asking

Considerate chitchat requires a reciprocal asking and answering of questions – asking too few or too many can appear self-centered or nosy. We control question-asking in order to study these trade-offs.

To control question-asking with weighted decoding, we use the binary decoding feature $\texttt{is\_qn\_word}(w)$, which is equal to 1 if and only if the word w is in a pre-defined list of interrogative words (*how, what, when, where, which, who, whom, whose, why, ?*). We find this is a somewhat effective method to encourage or discourage questions, but with unintended side-effects: a negative weight can discourage valid non-question utterances that happen to contain interrogative words (such as *I'm learning **how** to knit*) and a positive weight can result in degenerate utterances (such as *What???????* or *Who? When? How?*).

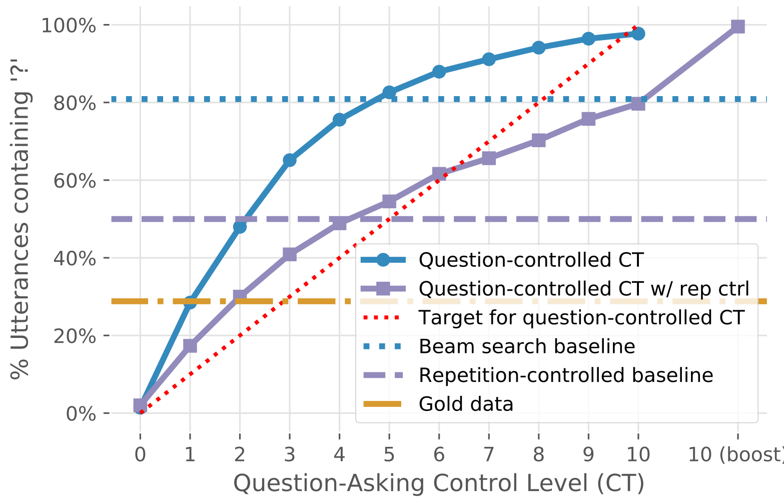


Figure 4.2: Controlling question-asking via conditional training. Exact numbers are provided in Table 4.6.

For conditional training, we regard an utterance y as containing a question if and only if y contains a question mark. We train our CT model on a control variable z with 11 possible values: $\{0, \ldots, 10\}$. As discussed in Section 4.5, we wish to control question-asking at the distributional, dialogue level, rather than at the binary, utterance level. Thus the setting $z = i$ means that the model should produce, on average, utterances containing '?' with probability $i/10$. During training we randomly assign examples to buckets such that each bucket i is trained on examples with the correct proportion of questions ($i/10$), and all buckets have the same amount of training examples.

We find that conditional training is effective to control question-asking – as shown in Figure 4.2, by increasing z from 0 to 10, we obtain a range of question-asking rates from 1.40% to 97.72%. However, when we introduce repetition control, question-asking is reduced – in particular, the $z = 10$ setting (which should produce 100% questions) now only produces 79.67% questions. The primary problem is the weighted decoding feature `extrep_bigram`, which discourages bigrams that have appeared in previous utterances – this prevents the model from producing bigrams that commonly occur in many questions, such as *do you* and *what is*. To fix this, we introduce an extra setting $z = 10$ *(boost)*, in which we do not use the feature `extrep_bigram` for weighted decoding during beam search, but we do use it to rerank the candidates after beam search. This setting, which allows the model to produce necessary question-asking bigrams, yields a 99.54% question-asking rate, at the cost of slightly increased external bigram repetition (see Table 4.6).

For controlling question-asking, conditional training is preferable to weighted decoding for two reasons. Firstly, it allows us to achieve (close to) 0% questions, 100% questions, or anything in between, without introducing the risk of degenerate output. Secondly, presence-of-a-question-mark captures the true attribute of interest (question-asking) more exactly and directly than presence of interrogative words. For these reasons, only the CT method is considered in the human evaluation.

4.7 Comparison of Control Methods

The previous section shows that conditional training and weighted decoding are both useful techniques, with different strengths and weaknesses.

The primary disadvantage of conditional training is that it sometimes fails to learn the connection between the control variable z and the target output y. In practice, we find the model can learn simple attributes of the output (such as the presence of '?', and overall genericness), but not relationships between the input and output (such as semantic relatedness). By contrast, weighted decoding can force the desired feature to appear in the output by raising the weight arbitrarily high (though this may have unintended side-effects).

The primary disadvantage of weighted decoding is that it risks going off-distribution when the weight is too strong. By contrast, conditional training produces mostly well-formed, in-distribution outputs. This highlights the importance of learned control – it is safer to learn to produce output that both satisfies the control variable and is appropriate, than to alter the decoding process to satisfy the control variable, potentially trading off appropriateness in the process.

Other considerations include: (1) Convenience: conditional training requires retraining; weighted decoding doesn't, but is slower at test time. (2) Data availability: conditional training requires training examples of the controllable attribute, whereas weighted decoding can control any computable feature without requiring examples. (3) Attribute definition: conditional training can control sentence-level attributes, but they must be discrete. By contrast, weighted decoding requires word-level features, but they can be continuous.

4.8 Human Evaluation Design

In order to study the effect of our controllable attributes, we conduct a large-scale human evaluation of 28 model configurations (see Table 4.5), plus human-human conversations for comparison.

In our evaluation, a crowdworker chats with a model (or in the human-human case, another crowdworker) for six conversational turns – the instructions and interface are shown in Figure 4.3. Afterwards, the crowdworker answers eight multiple-choice questions shown in Figure 4.4. These questions capture different aspects of conversational quality: avoiding repetition, interestingness, making sense, fluency, listening, inquisitiveness, humanness and engagingness. The eight questions are Likert questions on a 1–4 scale, where higher is better.[5] To match the ConvAI2 Challenge, we also add a persona retrieval question, in which the crowdworker is asked to select which of two possible personas was the model's persona.

Our evaluation is the same as the ConvAI2 Challenge evaluation, but more detailed – ConvAI2 includes only engagingness and persona retrieval.[6] As in the ConvAI2 challenge, each of our 28 model configurations was evaluated by over 100 crowdworkers, and the results were adjusted for annotator variance via a Bayesian calibration (Kulikov et al., 2019).

In designing our evaluation, we aimed to capture the four aspects we expected to directly improve via control (avoiding repetition, interestingness, listening, inquisitiveness), two important error classes we thought would be affected by our controls (fluency, making sense), and two overall quality measures (engagingness, humanness).

[5]Exceptions: Avoiding repetition is a 1–3 scale, as we found this gave clearer instructions. Inquisitiveness has an optimal score of 3; 1 and 2 represent too little question-asking, and 4 represents too much.

[6]There are three other minor differences between our evaluation and ConvAI2's: (1) We fix capitalization and spacing before showing the chatbot's utterances to crowdworkers, while ConvAI2 show the raw lowercase tokenized form. We found the latter interferes with fluency evaluation. (2) We conduct 6 dialogue turns, while ConvAI2 conducts 4-6. This was necessary to evaluate repetitiveness. (3) We use (publicly-available) validation set personas, while ConvAI2 uses (hidden) test set personas. This enables us to release our evaluation chatlogs.

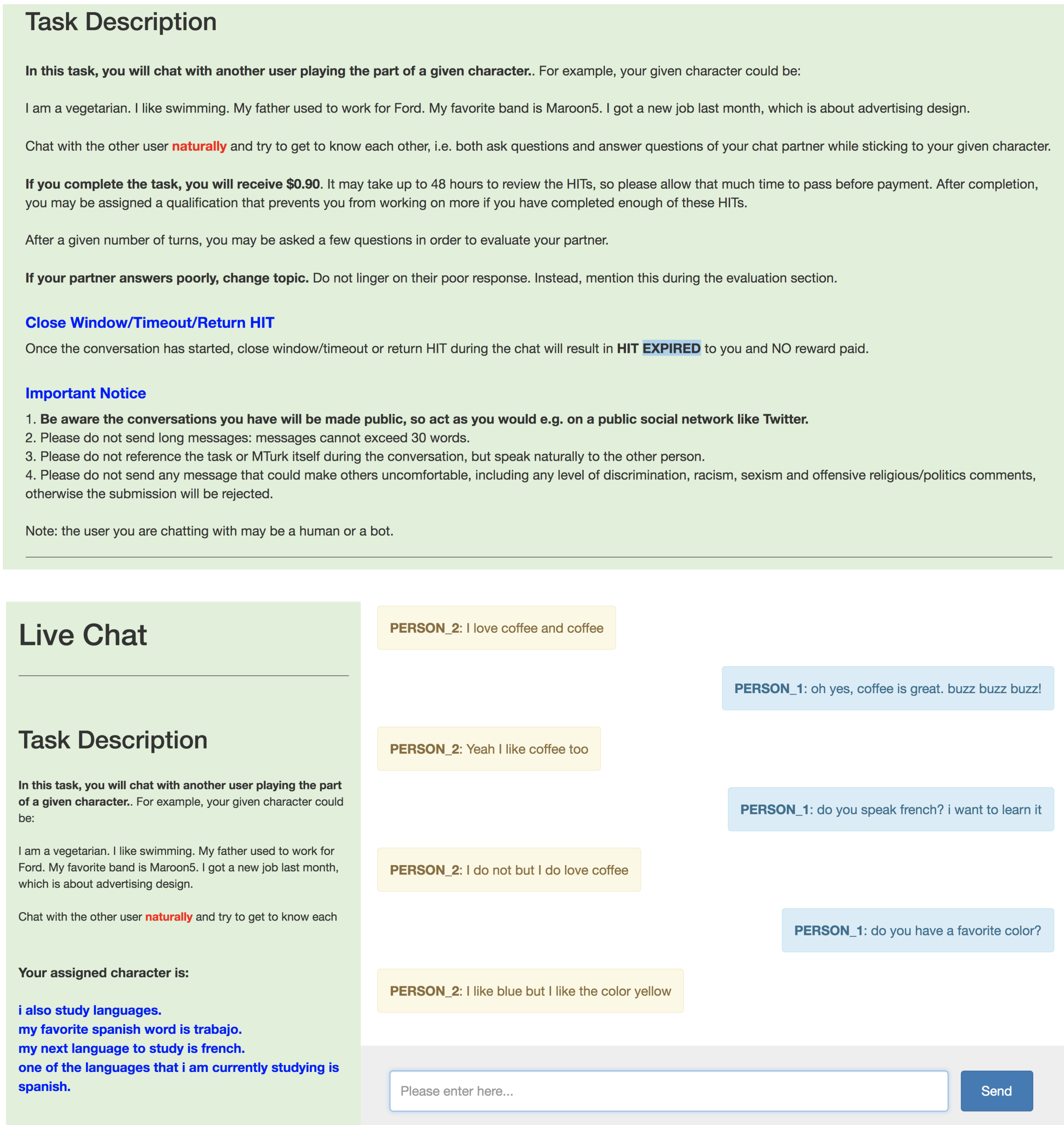


Figure 4.3: Above: screenshot of the task description. Below: screenshot of the chat UI, talking with the beam search baseline model.

[Engagingness] How much did you enjoy talking to this user?
• Not at all • A little • Somewhat • A lot

[Interestingness] How interesting or boring did you find this conversation?
• Very boring • A little boring • A little interesting • Very interesting

[Inquisitiveness] How much did the user try to get to know you?
• Didn't ask about me at all • Asked about me some
• Asked about me a good amount • Asked about me too much

[Listening] How much did the user seem to pay attention to what you said?
• Always ignored what I said • Mostly ignored what I said
• Mostly paid attention to what I said • Always paid attention to what I said

[Avoiding Repetition] How repetitive was this user?
• Repeated themselves over and over • Sometimes said the same thing twice
• Always said something new

[Fluency] How naturally did this user speak English?
• Very unnatural • Mostly unnatural • Mostly natural • Very natural

[Making sense] How often did this user say something which did NOT make sense?
• Never made any sense • Most responses didn't make sense
• Some responses didn't make sense • Everything made perfect sense

[Humanness] Do you think this user is a bot or a human?
• Definitely a bot • Probably a bot • Probably a human • Definitely a human

[Persona retrieval] Which prompt (character) do you think the other user was given for this conversation?
Respondent chooses one of two provided personas

Figure 4.4: The questions and multiple-choice options used in the human evaluation , in the order presented.

4.9 Human Evaluation Results

4.9.1 Main Findings

In this section we summarize the main findings of our human evaluation – whose full results can be found in Section 4.11 and Figure 4.11, with sample conversations in Figure 4.10.

As Figure 4.5 shows, controlling for repetition, specificity and question-asking all lead to large engagingness improvements over the greedy and beam-search baseline models. In particular, we find that controlling for multi-turn (self) repetition is important and should

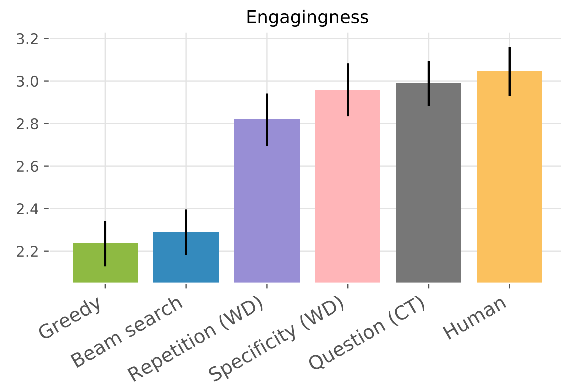


Figure 4.5: Calibrated human judgments of engagingness for the baselines and best controlled models. Note: In Figures 4.6, 4.7, 4.8 and here, the Specificity and Question controlled models both include Repetition control, but Question control doesn't include Specificity control, or vice versa.

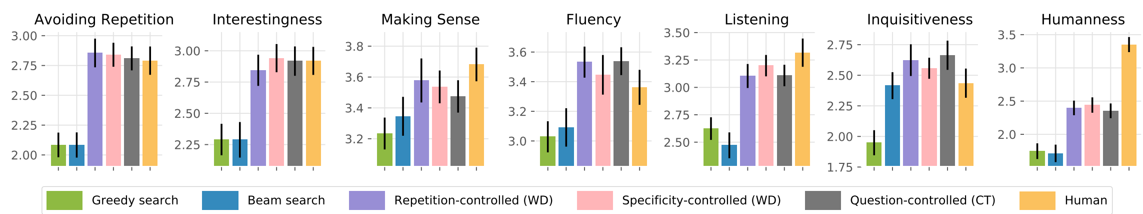


Figure 4.6: Calibrated human judgments of conversational aspects for the baselines and best controlled models.

be incorporated alongside other attribute control methods. We found no improvement by controlling response-relatedness.

To better understand these overall engagingness improvements, we consider the full set of human judgments, shown in Figure 4.6. We find that reducing repetition leads to improvements across all our aspects of conversational quality. Increasing specificity shows improvements in interestingness and listening ability over the repetition-controlled baseline, while increasing question-asking shows improvements in inquisitiveness and interestingness over the repetition-controlled baseline.

Our most engaging model, which controls both repetition and question-asking – marked 'Question (CT)' in Figure 4.5 – matches the engagingness of the winning entry in the

ConvAI2 competition, as both models achieve a raw score[7] of 3.1 (Dinan et al., 2019a). However, the ConvAI2 winner, Lost in Conversation, was trained on approximately 12× as much data as our model. Lost in Conversation is based on the OpenAI GPT Language Model (Radford et al., 2018), which is pretrained on the BookCorpus (Zhu et al., 2015), which contains approximately 985 million words, whereas our model is pretrained on the Twitter dataset (approximately 79 million words).

Altogether, our evaluation clearly shows that controlling low-level attributes over multiple turns leads to improved overall quality.

4.9.2 Effect of Controlled Attributes

Repetition (WD) We observe that self-repetition across utterances (*external repetition*) is by far the most severe form of repetition in our beam search baseline model. We evaluate several settings of the `extrep_bigram` weighted decoding feature, and find that an aggressive repetition-reduction setting (reducing bigram repetition rate to below gold data levels) is rated best. We also find that blocking repeated content words improves the avoiding repetition score. See Tables 4.5, 4.6 and Section 4.11 for full details.

As shown in Figure 4.5 and Figure 4.6, our repetition-controlled model improves hugely over the beam search baseline in all metrics, and achieves close-to-human scores on all metrics except humanness. This striking result demonstrates that repetition is by far the biggest limiting quality factor for naive sequence-to-sequence dialogue agents. The result also emphasizes the importance of *multi-turn* dialogue evaluation to detect the problem. We refer to this model as the *repetition-controlled baseline*, and use it as a basis for all remaining experiments (i.e., we control specificity, response-relatedness and question-asking on top of these repetition-control settings).

Specificity (WD, CT) For our weighted decoding models, the extreme settings (very generic and very specific) score poorly in engagingness due to the frequent presence of degenerate output – see Figure 4.7. We find that the weight $= 4$ setting (which is more

[7] Although the same Bayesian calibration method was applied both in our study and in the ConvAI2 competition, calibrated scores are not comparable across the two; thus we compare raw scores (viewable in Table 4.7).

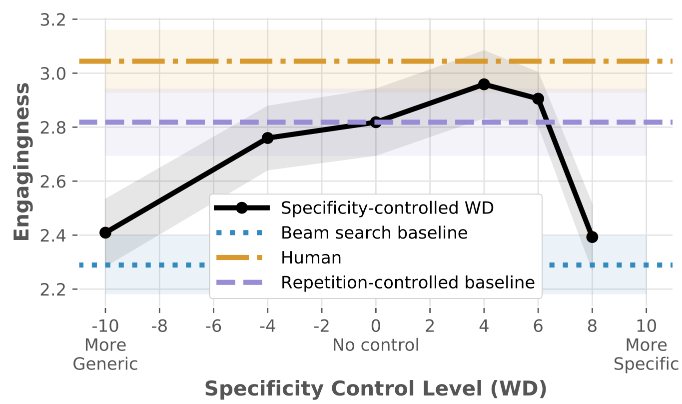


Figure 4.7: Calibrated human judgments of engagingness for different specificity control settings.

specific than the repetition-controlled baseline and about as specific as the gold data) maximizes engagingness. As shown in Figure 4.5 and Figure 4.6, this more-specific model is rated more interesting, engaging, and a better listener than the repetition-controlled baseline, but at the cost of reduced fluency and making sense. Our CT model with $z = 7$ (which has a similar NIDF level as WD with weight $= 4$) shows similar results, but the improvements are smaller. For further discussion on the interestingness of our specificity models, see Section 4.9.3.

Response-relatedness (WD) We evaluated several control settings (weight = -10, 5, 10, 13) and found that none scored better than weight $= 0$ (no response-relatedness control); see Figure 4.11. This is surprising – prior to running the human evaluation, we annotated 100 examples ourselves to determine the best control settings. While we identified a more responsive setting (weight $= 5$) as less likely than the uncontrolled model to ignore the user, crowdworkers rated it as a slightly *worse* listener than the uncontrolled model. One explanation for this discrepancy is that the more responsive model takes more risks, using more rare words (0.197 NIDF, up from 0.178), and thus receives a lower makes-sense score (3.41, down from 3.70). We hypothesize that, compared to us, the crowdworkers are less tolerant of slightly nonsensical output, and more tolerant of generic unrelated utterances.

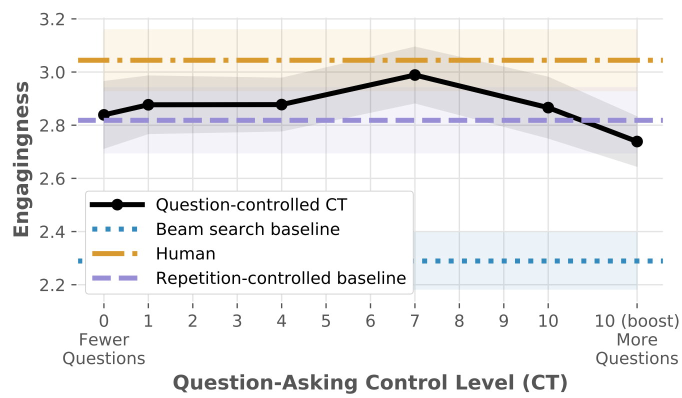


Figure 4.8: Calibrated human judgments of engagingness for different question-asking control settings.

Question-asking (CT) As shown in Figure 4.8, a question-asking rate of 65.7% ($z = 7$) maximizes engagingness. This setting, which asks more questions than both the repetition-controlled baseline (50.0%) and the human-produced gold data (28.8%), brings us closest to human-level engagingness – see Figure 4.5. Although we find that a rate of approximately 65.7% question-asking is the most engaging, a lower level (48.9%, or $z = 4$) is rated the best listener. Lastly, we find that although asking too many questions is less engaging, most crowdworkers will not directly criticize a chatbot that asks questions on every turn – only 11.9% of crowdworkers judged the $z = 10$ *(boost)* setting, which asks 99.5% questions, as asking too many questions.[8] For full details of these scores, see Table 4.6 and Figure 4.11.

For time and budget reasons, we did not evaluate any models controlling both question-asking and specificity. However, we expect it is possible to obtain further improvements by doing so.

4.9.3 A/B Tests for Interestingness

Though our more-specific models yielded significant improvements in engagingness, we were surprised that they did not yield clearer improvements in interestingness. To investigate further, we conducted an A/B interestingness evaluation of three specificity-controlled

[8]Though this conclusion may hold true for the PersonaChat task – a synthetic chatting task that instructs participants to get to know each other – in real-life social conversations, incessant question-asking is less appropriate (Hardy et al., 2021).

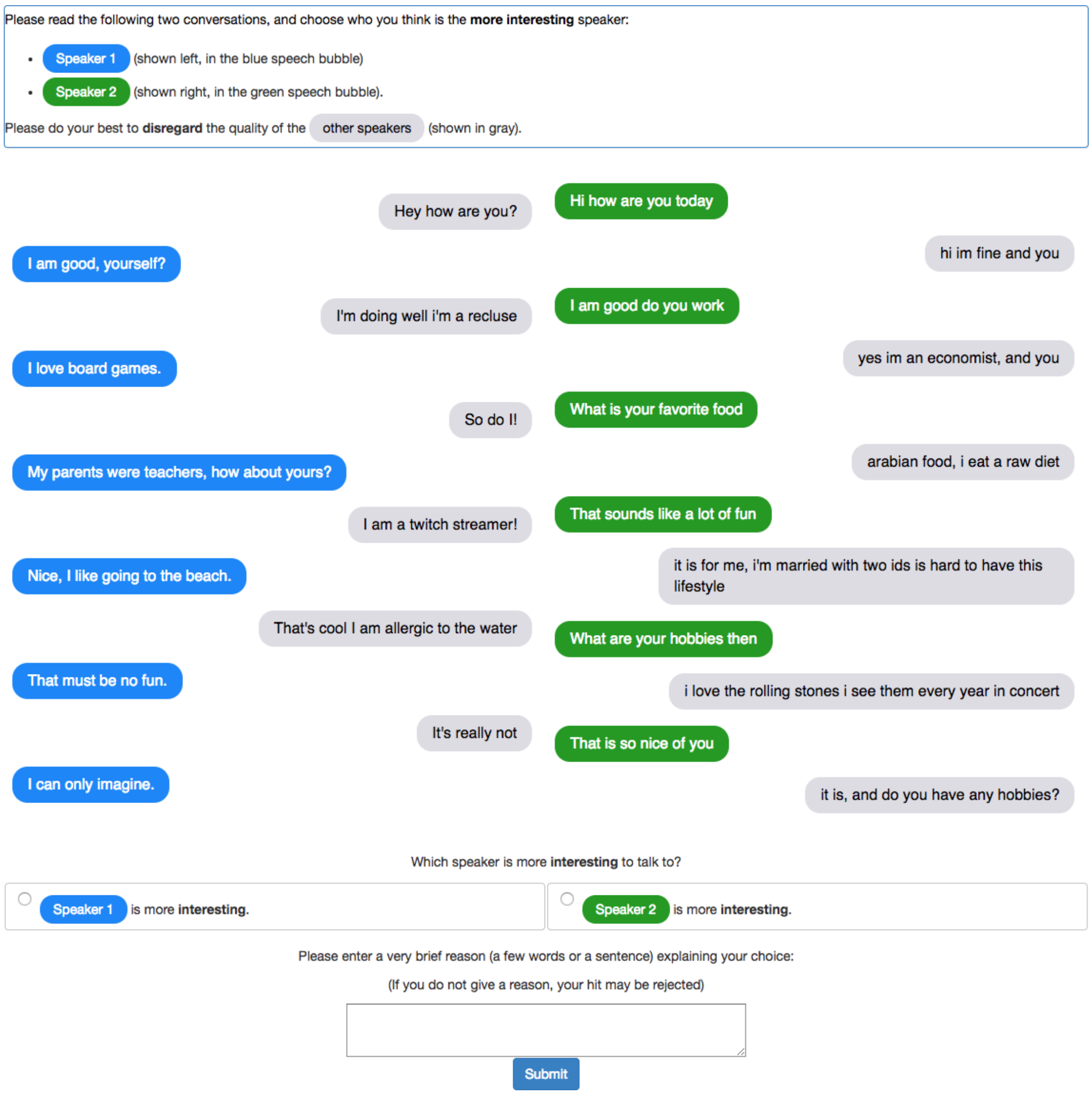


Figure 4.9: Screenshot of the A/B test UI , comparing a human-human conversation (left) and a Repetition-controlled baseline model (right).

Model	**Win%**	**Top 3 reasons for preferring model**
Specificity WD (weight = 6)	84.1%	*More information; Better flow; More descriptive*
Specificity WD (weight = 4)	75.5%	*More information; They describe their life in more detail; Funny*
Specificity CT ($z = 7$)	56.2%	*More information; Better flow; Seems more interested*

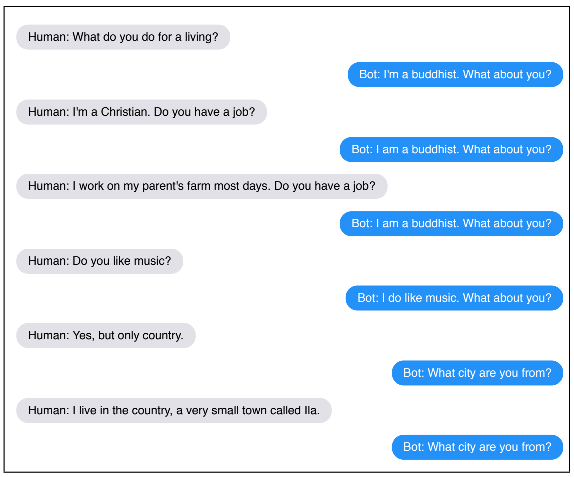
Table 4.4: A/B tests comparing various specificity-controlled models to the repetition-controlled baseline on interestingness. We find all comparisons are significant ($p < .05$; binomial test).

models, compared to the repetition-controlled baseline. Crowdworkers were shown two conversations (from the main human evaluation) and asked to choose which model was more interesting (see Figure 4.9 for details). We collected 500 samples per comparison, plus 200 additional human vs repetition-controlled baseline samples, which were used to filter for quality control. After discarding low-quality crowdworkers, we have roughly 300 evaluations per comparison, with an average Cohen's $\kappa = 0.6$.

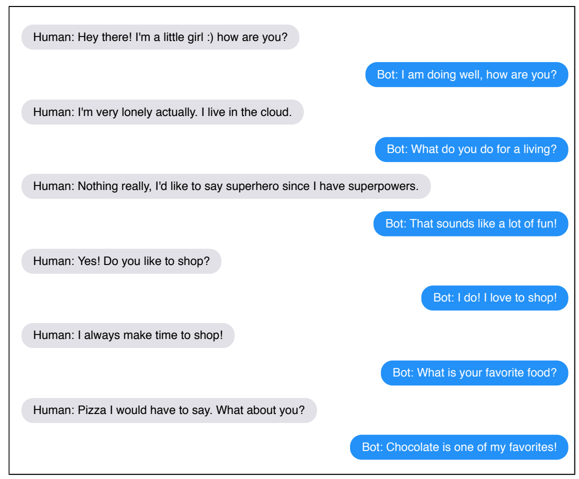
As shown in Table 4.4, all three models were rated significantly more interesting than the repetition-controlled baseline. This convincingly shows that producing utterances with more rare words is a valid strategy to improve interestingness. We have two explanations for why these interestingness differences did not materialize in our main evaluation. Firstly, interestingness is a particularly subjective metric (unlike more tangible metrics such as avoiding repetition and making sense) – this makes it hard to calibrate across crowdworkers. Secondly, we suspect that in our original evaluation, the crowdworkers may have evaluated the interestingness of the *task* rather than the *chatbot*. This could account for why subtle increases in conversational ability did not result in higher interestingness ratings – the PersonaChat task itself has a natural interestingness limit.

4.10 Example Crowdworker-Bot Conversations

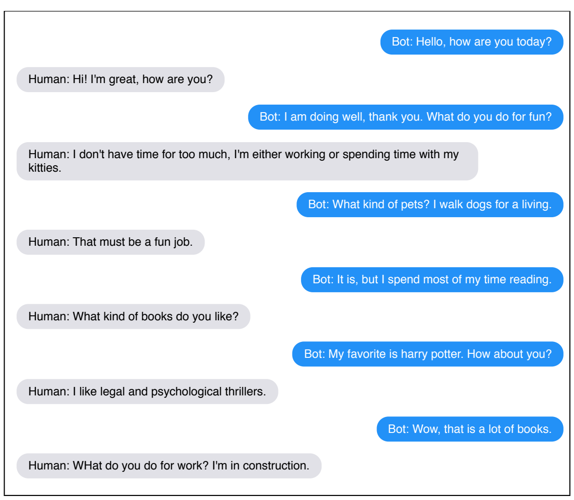
Figure 4.10 shows how controlling repetition, question-asking and specificity can improve conversations.



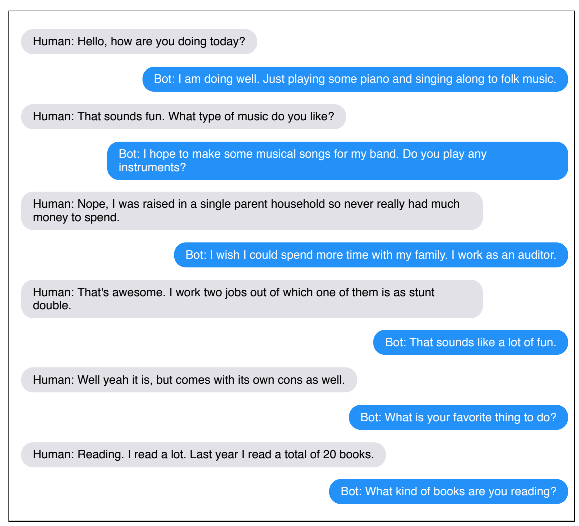
(a)



(b)



(c)



(d)

Figure 4.10: Example conversations with (a) Baseline (b) Repetition-controlled baseline (c) Question-controlled CT ($z = 7$), (d) Specificity-controlled WD (weight $= 4$).

4.11 Tables of Settings and Results for All Configurations

This section contains details of the exact control settings we used, and the automatic and human evaluation results achieved, by all model configurations.

	Repetition					Specificity	Response-rel	Questions
	External		Internal		Partner Rep.			
	Bigram	Unigram	Bigram	Unigram	Bigram	NIDF	Cos sim	Has '?'
Baselines								
Greedy Search								
Beam Search (beam size 20)								
Repetition control (WD)								
Extrep bigram WD -0.5	wt -0.5							
Extrep bigram WD -1.25	wt -1.25							
Extrep bigram WD -3.5	wt -3.5							
Extrep bigram WD -inf	wt -∞							
Repetition-controlled baseline	wt -3.5	wt -∞		wt -∞				
Question control (CT)								
Question-controlled CT 0	wt -3.5	wt -∞		wt -∞				$z = 0$
Question-controlled CT 1	wt -3.5	wt -∞		wt -∞				$z = 1$
Question-controlled CT 4	wt -3.5	wt -∞		wt -∞				$z = 4$
Question-controlled CT 7	wt -3.5	wt -∞		wt -∞				$z = 7$
Question-controlled CT 10	wt -3.5	wt -∞		wt -∞				$z = 10$
Question-controlled CT 10 (boost)	wt 0*	wt -∞		wt -∞				$z = 10$
Specificity control (CT)								
Specificity-controlled CT 0	wt -3.5	wt -∞		wt -∞		$z = 0$		
Specificity-controlled CT 2	wt -3.5	wt -∞		wt -∞		$z = 2$		
Specificity-controlled CT 4	wt -3.5	wt -∞		wt -∞		$z = 4$		
Specificity-controlled CT 7	wt -3.5	wt -∞		wt -∞		$z = 7$		
Specificity-controlled CT 9	wt -3.5	wt -∞		wt -∞		$z = 9$		
Specificity control (WD)								
Specificity-controlled WD -10	wt -3.5	wt -∞		wt -∞		wt -10		
Specificity-controlled WD -4	wt -3.5	wt -∞		wt -∞		wt -4		
Specificity-controlled WD 4	wt -3.5	wt -∞		wt -∞		wt 4		
Specificity-controlled WD 6	wt -3.5	wt -∞		wt -∞		wt 6		
Specificity-controlled WD 8	wt -3.5	wt -∞		wt -∞		wt 8		
Response-related control (WD) **								
Response-related controlled WD -10	wt -3.5	wt -∞	wt -∞	wt -∞	wt -∞		wt -10	
Response-related controlled WD 0	wt -3.5	wt -∞	wt -∞	wt -∞	wt -∞		wt 0	
Response-related controlled WD 5	wt -3.5	wt -∞	wt -∞	wt -∞	wt -∞		wt 5	
Response-related controlled WD 10	wt -3.5	wt -∞	wt -∞	wt -∞	wt -∞		wt 10	
Response-related controlled WD 13	wt -3.5	wt -∞	wt -∞	wt -∞	wt -∞		wt 13	

Table 4.5: Control settings for all configurations. 'wt' means a weighted decoding (WD) feature weight and z means the control variable setting in conditional training (CT).
* In Question-controlled CT 10 (boost), the feature `extrep_bigram` is *not* used for WD during beam search, but *is* used to rerank candidates after beam search; see Section 4.6.4.
** Note that the Response-related controlled models additionally introduce repetition controls to block internal bigram repetition and partner bigram repetition. This was necessary to prevent the model from parroting the partner's last utterance. In Table 4.8, we find that just adding these extra repetition controls (here called Response-related controlled WD 0, i.e., increased repetition control but no response-relatedness control) outperforms our canonical Repetition-controlled baseline. However, given that we discovered this later, our specificity and question controlled models are built on top of the canonical Repetition-controlled baseline.

	Repetition					Specificity	Response-rel	Questions
	External		Internal		Partner Rep.			
	Bigram	Unigram	Bigram	Unigram	Bigram	NIDF	Cos sim	Has '?'
Gold data and baselines								
Gold Data	4.65%	9.62%	0.38%	0.97%	5.10%	0.2119	0.1691	28.80%
Greedy Search	35.88%	36.31%	8.08%	10.59%	12.20%	0.1688	0.1850	6.46%
Beam Search (beam size 20)	46.85%	44.15%	0.32%	0.61%	12.90%	0.1662	0.0957	80.87%
Repetition control (WD)								
Extrep bigram WD -0.5	19.70%	16.85%	0.26%	0.62%	11.93%	0.1730	0.1348	73.04%
Extrep bigram WD -1.25	4.62%	4.79%	0.40%	0.89%	10.61%	0.1763	0.1504	61.22%
Extrep bigram WD -3.5	0.75%	4.61%	0.47%	0.94%	9.89%	0.1771	0.1681	48.89%
Extrep bigram WD -inf	0.00%	4.74%	0.51%	1.05%	9.56%	0.1780	0.1711	45.98%
Repetition-controlled baseline	0.73%	0.00%	0.17%	0.00%	9.55%	0.1766	0.1676	49.98%
Question control (CT)								
Question-controlled CT 0	0.06%	0.00%	0.19%	0.00%	9.20%	0.1871	0.1753	2.01%
Question-controlled CT 1	0.09%	0.00%	0.19%	0.00%	8.66%	0.1844	0.1722	17.33%
Question-controlled CT 4	0.40%	0.00%	0.25%	0.00%	8.53%	0.1794	0.1713	48.88%
Question-controlled CT 7	0.80%	0.00%	0.17%	0.00%	8.48%	0.1771	0.1724	65.65%
Question-controlled CT 10	1.27%	0.00%	0.16%	0.00%	8.48%	0.1761	0.1728	79.67%
Question-controlled CT 10 (boost)	7.64%	0.00%	0.03%	0.00%	10.76%	0.1701	0.1651	99.54%
Specificity control (CT)								
Specificity-controlled CT 0	0.60%	0.00%	0.20%	0.00%	9.05%	0.1478	0.1522	48.75%
Specificity-controlled CT 2	0.28%	0.00%	0.10%	0.00%	8.37%	0.1772	0.1833	50.57%
Specificity-controlled CT 4	0.12%	0.00%	0.08%	0.00%	7.90%	0.1921	0.1877	29.46%
Specificity-controlled CT 7	0.02%	0.00%	0.14%	0.00%	8.17%	0.2156	0.1955	16.51%
Specificity-controlled CT 9	0.01%	0.00%	0.11%	0.00%	8.01%	0.2462	0.1990	8.50%
Specificity control (WD)								
Specificity-controlled WD -10	0.14%	0.00%	10.59%	0.00%	8.70%	0.1107	0.0994	33.55%
Specificity-controlled WD -4	0.65%	0.00%	1.98%	0.00%	9.95%	0.1501	0.1398	44.92%
Specificity-controlled WD 4	0.15%	0.00%	0.19%	0.00%	7.54%	0.2121	0.1972	45.53%
Specificity-controlled WD 6	0.07%	0.00%	0.13%	0.00%	6.50%	0.2546	0.2040	39.37%
Specificity-controlled WD 8	0.01%	0.00%	0.10%	0.00%	3.40%	0.4035	0.1436	26.68%
Response-related control (WD)								
Response-related controlled WD -10	0.13%	0.00%	0.00%	0.00%	0.00%	0.1914	-0.0921	25.71%
Response-related controlled WD 0	0.24%	0.00%	0.00%	0.00%	0.00%	0.1785	0.1414	44.55%
Response-related controlled WD 5	0.15%	0.00%	0.00%	0.00%	0.00%	0.1973	0.4360	39.78%
Response-related controlled WD 10	0.05%	0.00%	0.00%	0.00%	0.00%	0.2535	0.6653	27.56%
Response-related controlled WD 13	0.02%	0.00%	0.00%	0.00%	0.00%	0.2999	0.7251	20.47%

Table 4.6: Automatic metrics (computed over validation set) for all model configurations that were human-evaluated.

Model	Avoiding Rep.	Engage	Fluency	Humanness	Inquisitive	Interesting	Listening	Make Sense	Persona
Human and baselines									
Human	2.90 ± 0.39	3.31 ± 0.90	3.66 ± 0.71	3.40 ± 0.80	2.63 ± 0.63	3.23 ± 0.83	3.64 ± 0.63	3.84 ± 0.52	0.92 ± 0.27
Greedy Search	2.16 ± 0.72	2.31 ± 1.08	3.20 ± 0.81	1.78 ± 0.90	2.00 ± 0.81	2.36 ± 0.98	2.78 ± 0.84	3.33 ± 0.75	0.87 ± 0.34
Beam Search (beam size 20)	2.14 ± 0.72	2.35 ± 1.01	3.23 ± 0.93	1.81 ± 0.87	2.50 ± 0.72	2.35 ± 0.98	2.63 ± 0.85	3.40 ± 0.77	0.77 ± 0.42
Repetition control (WD)									
Extrep bigram WD -0.5	2.66 ± 0.56	2.56 ± 0.92	3.57 ± 0.64	2.19 ± 0.94	2.67 ± 0.62	2.61 ± 0.87	3.08 ± 0.78	3.60 ± 0.57	0.75 ± 0.43
Extrep bigram WD -1.25	2.84 ± 0.39	2.91 ± 0.90	3.59 ± 0.64	2.32 ± 0.98	2.63 ± 0.60	2.86 ± 0.89	3.21 ± 0.71	3.64 ± 0.62	0.72 ± 0.45
Extrep bigram WD -3.5	2.90 ± 0.30	2.95 ± 0.86	**3.73 ± 0.50**	2.45 ± 1.03	2.55 ± 0.61	2.88 ± 0.80	3.27 ± 0.79	3.68 ± 0.49	0.80 ± 0.40
Extrep bigram WD -inf	2.82 ± 0.43	2.96 ± 0.86	3.64 ± 0.58	2.40 ± 0.96	2.65 ± 0.69	2.86 ± 0.82	3.31 ± 0.69	3.66 ± 0.59	0.91 ± 0.29
Repetition-controlled baseline	2.89 ± 0.39	2.89 ± 0.89	3.66 ± 0.56	2.50 ± 0.99	2.70 ± 0.64	2.96 ± 0.92	3.25 ± 0.71	3.68 ± 0.54	0.87 ± 0.34
Question control (CT)									
Question-controlled CT 0	2.95 ± 0.25	2.92 ± 0.90	3.70 ± 0.54	2.49 ± 0.97	2.48 ± 0.72	2.85 ± 0.93	3.29 ± 0.69	3.56 ± 0.66	0.86 ± 0.35
Question-controlled CT 1	2.88 ± 0.33	2.94 ± 0.93	3.59 ± 0.66	2.47 ± 0.95	2.52 ± 0.69	2.85 ± 0.90	3.32 ± 0.73	3.63 ± 0.55	0.85 ± 0.36
Question-controlled CT 4	2.88 ± 0.38	2.88 ± 0.94	3.59 ± 0.73	2.42 ± 1.07	2.55 ± 0.66	2.82 ± 0.85	**3.37 ± 0.74**	3.63 ± 0.59	0.84 ± 0.37
Question-controlled CT 7	2.88 ± 0.37	**3.07 ± 0.90**	3.67 ± 0.54	2.42 ± 0.98	2.75 ± 0.58	2.97 ± 0.84	3.23 ± 0.76	3.53 ± 0.76	0.80 ± 0.40
Question-controlled CT 10	2.74 ± 0.46	2.90 ± 0.93	3.70 ± 0.50	2.43 ± 1.04	2.71 ± 0.57	2.72 ± 0.88	3.12 ± 0.73	3.59 ± 0.66	0.79 ± 0.41
Question-controlled CT 10 (boost)	2.76 ± 0.49	2.84 ± 0.94	3.60 ± 0.64	2.26 ± 0.97	**2.94 ± 0.57**	2.83 ± 0.94	3.18 ± 0.80	3.52 ± 0.67	0.72 ± 0.45
Specificity control (CT)									
Specificity-controlled CT 0	2.83 ± 0.40	2.96 ± 0.93	3.62 ± 0.58	2.42 ± 0.99	2.60 ± 0.56	2.86 ± 0.89	3.29 ± 0.70	3.66 ± 0.60	0.72 ± 0.45
Specificity-controlled CT 2	2.90 ± 0.36	2.78 ± 1.00	3.60 ± 0.64	2.37 ± 0.93	2.66 ± 0.66	2.80 ± 0.96	3.14 ± 0.77	3.50 ± 0.63	0.81 ± 0.39
Specificity-controlled CT 4	2.92 ± 0.27	2.81 ± 0.88	3.65 ± 0.59	2.34 ± 1.02	2.57 ± 0.62	2.80 ± 0.78	3.25 ± 0.78	3.50 ± 0.66	0.86 ± 0.35
Specificity-controlled CT 7	2.89 ± 0.32	3.00 ± 0.94	3.64 ± 0.67	2.53 ± 1.03	2.56 ± 0.66	2.90 ± 0.90	3.34 ± 0.70	3.59 ± 0.60	0.82 ± 0.39
Specificity-controlled CT 9	2.90 ± 0.35	2.83 ± 0.87	3.61 ± 0.62	2.40 ± 0.97	2.31 ± 0.74	2.84 ± 0.83	3.07 ± 0.81	3.58 ± 0.56	0.88 ± 0.32
Specificity control (WD)									
Specificity-controlled WD -10	2.85 ± 0.43	2.43 ± 0.99	3.34 ± 0.83	2.15 ± 0.91	2.31 ± 0.69	2.38 ± 0.94	3.03 ± 0.75	3.33 ± 0.70	0.71 ± 0.45
Specificity-controlled WD -4	2.90 ± 0.30	2.78 ± 0.95	3.55 ± 0.63	2.41 ± 0.92	2.52 ± 0.66	2.64 ± 0.93	3.28 ± 0.73	3.56 ± 0.62	0.82 ± 0.38
Specificity-controlled WD 4	2.95 ± 0.21	2.99 ± 0.86	3.65 ± 0.55	2.49 ± 0.90	2.65 ± 0.55	3.00 ± 0.78	**3.37 ± 0.59**	3.63 ± 0.50	**0.93 ± 0.25**
Specificity-controlled WD 6	2.93 ± 0.26	2.96 ± 0.90	3.52 ± 0.76	2.41 ± 1.04	2.58 ± 0.66	**3.06 ± 0.80**	3.24 ± 0.76	3.50 ± 0.66	**0.93 ± 0.26**
Specificity-controlled WD 8	2.78 ± 0.52	2.40 ± 1.23	2.67 ± 1.25	1.86 ± 0.97	2.03 ± 0.87	2.55 ± 1.14	2.61 ± 1.05	2.91 ± 0.91	0.92 ± 0.28
Response-related control (WD)									
Response-related controlled WD -10	2.86 ± 0.44	2.48 ± 0.98	3.42 ± 0.74	2.02 ± 0.93	2.38 ± 0.75	2.53 ± 0.94	2.84 ± 0.80	3.14 ± 0.75	0.91 ± 0.29
Response-related controlled WD 0	**2.96 ± 0.23**	3.01 ± 0.90	3.72 ± 0.54	**2.73 ± 1.00**	2.56 ± 0.67	2.92 ± 0.84	**3.37 ± 0.72**	**3.73 ± 0.52**	0.82 ± 0.38
Response-related controlled WD 5	2.90 ± 0.33	2.88 ± 0.90	3.51 ± 0.63	2.41 ± 1.01	2.53 ± 0.65	2.85 ± 0.90	3.27 ± 0.73	3.49 ± 0.63	0.82 ± 0.39
Response-related controlled WD 10	2.78 ± 0.43	2.39 ± 1.04	3.06 ± 0.90	1.97 ± 0.99	2.22 ± 0.67	2.57 ± 1.01	3.03 ± 0.76	3.16 ± 0.63	0.75 ± 0.43
Response-related controlled WD 13	2.71 ± 0.57	2.10 ± 1.13	2.54 ± 1.12	1.81 ± 1.07	2.14 ± 0.84	2.33 ± 1.06	2.69 ± 0.83	2.70 ± 0.88	0.62 ± 0.49

Table 4.7: Raw scores (mean ± std.) for all models and human evaluation metrics.

The first eight columns are Likert metrics on a 1–4 scale (except Avoiding Repetition, which is a 1–3 scale), where higher is better (except Inquisitiveness, which has an optimal score of 3). The last column, Persona Retrieval, is on a scale from 0 to 1 where higher is better.

The maximum of each column (excluding Human row) is in bold.

Model	Avoiding Rep.	Engage	Fluency	Humanness	Inquisitive	Interesting	Listening	Make Sense
Human and baselines								
* Human	2.79 ± 0.12	3.04 ± 0.11	3.36 ± 0.12	3.35 ± 0.11	2.44 ± 0.12	2.92 ± 0.11	3.32 ± 0.13	3.68 ± 0.11
* Greedy Search	2.08 ± 0.10	2.24 ± 0.11	3.03 ± 0.10	1.75 ± 0.12	1.95 ± 0.10	2.29 ± 0.13	2.62 ± 0.10	3.23 ± 0.10
* Beam Search (beam size 20)	2.08 ± 0.11	2.29 ± 0.11	3.09 ± 0.13	1.71 ± 0.13	2.42 ± 0.11	2.29 ± 0.14	2.47 ± 0.12	3.35 ± 0.13
Repetition control (WD)								
Extrep bigram WD -0.5	2.62 ± 0.10	2.54 ± 0.12	3.35 ± 0.12	2.13 ± 0.11	2.63 ± 0.11	2.56 ± 0.11	2.93 ± 0.11	3.48 ± 0.11
Extrep bigram WD -1.25	2.78 ± 0.09	2.82 ± 0.13	3.40 ± 0.12	2.27 ± 0.12	2.54 ± 0.09	2.76 ± 0.10	3.05 ± 0.11	3.53 ± 0.14
Extrep bigram WD -3.5	2.83 ± 0.11	2.93 ± 0.10	**3.56 ± 0.10**	2.43 ± 0.11	2.47 ± 0.11	2.83 ± 0.10	3.14 ± 0.10	3.62 ± 0.12
Extrep bigram WD -inf	2.74 ± 0.11	2.87 ± 0.14	3.49 ± 0.12	2.32 ± 0.13	2.56 ± 0.11	2.75 ± 0.12	3.13 ± 0.12	3.59 ± 0.12
* Repetition-controlled baseline	2.86 ± 0.12	2.82 ± 0.12	3.53 ± 0.10	2.40 ± 0.11	2.62 ± 0.13	2.84 ± 0.12	3.10 ± 0.11	3.58 ± 0.14
Question control (CT)								
Question-controlled CT 0	**2.87 ± 0.12**	2.84 ± 0.13	3.51 ± 0.10	2.46 ± 0.11	2.36 ± 0.09	2.76 ± 0.09	3.10 ± 0.10	3.49 ± 0.12
Question-controlled CT 1	2.82 ± 0.11	2.88 ± 0.11	3.42 ± 0.10	2.46 ± 0.12	2.47 ± 0.11	2.79 ± 0.13	3.14 ± 0.11	3.55 ± 0.10
Question-controlled CT 4	2.78 ± 0.12	2.88 ± 0.10	3.47 ± 0.11	2.40 ± 0.09	2.53 ± 0.13	2.83 ± 0.13	**3.24 ± 0.11**	3.59 ± 0.10
* Question-controlled CT 7	2.81 ± 0.10	**2.99 ± 0.11**	3.54 ± 0.09	2.35 ± 0.11	2.66 ± 0.12	2.92 ± 0.12	3.11 ± 0.10	3.47 ± 0.10
Question-controlled CT 10	2.67 ± 0.13	2.87 ± 0.11	3.52 ± 0.12	2.35 ± 0.12	2.63 ± 0.12	2.66 ± 0.10	2.94 ± 0.11	3.53 ± 0.12
Question-controlled CT 10 (boost)	2.68 ± 0.12	2.74 ± 0.09	3.42 ± 0.12	2.19 ± 0.13	**2.79 ± 0.11**	2.74 ± 0.11	3.00 ± 0.12	3.45 ± 0.13
Specificity control (CT)								
Specificity-controlled CT 0	2.79 ± 0.10	2.93 ± 0.09	3.44 ± 0.12	2.38 ± 0.11	2.56 ± 0.12	2.84 ± 0.12	3.12 ± 0.13	3.61 ± 0.11
Specificity-controlled CT 2	2.78 ± 0.12	2.74 ± 0.11	3.39 ± 0.13	2.31 ± 0.13	2.56 ± 0.13	2.74 ± 0.12	2.99 ± 0.11	3.47 ± 0.10
Specificity-controlled CT 4	2.82 ± 0.10	2.80 ± 0.13	3.44 ± 0.14	2.32 ± 0.13	2.51 ± 0.12	2.78 ± 0.15	3.09 ± 0.13	3.46 ± 0.13
Specificity-controlled CT 7	2.81 ± 0.12	2.91 ± 0.13	3.43 ± 0.11	2.45 ± 0.10	2.49 ± 0.11	2.81 ± 0.12	3.15 ± 0.12	3.55 ± 0.11
Specificity-controlled CT 9	2.80 ± 0.13	2.78 ± 0.10	3.41 ± 0.12	2.35 ± 0.13	2.28 ± 0.11	2.79 ± 0.11	2.91 ± 0.11	3.51 ± 0.12
Specificity control (WD)								
Specificity-controlled WD -10	2.76 ± 0.11	2.41 ± 0.12	3.19 ± 0.12	2.15 ± 0.11	2.28 ± 0.13	2.35 ± 0.12	2.89 ± 0.11	3.28 ± 0.12
Specificity-controlled WD -4	2.83 ± 0.10	2.76 ± 0.12	3.37 ± 0.10	2.36 ± 0.11	2.46 ± 0.11	2.62 ± 0.12	3.14 ± 0.09	3.52 ± 0.11
* Specificity-controlled WD 4	2.84 ± 0.10	2.96 ± 0.12	3.45 ± 0.13	2.44 ± 0.12	2.56 ± 0.09	**2.94 ± 0.11**	3.20 ± 0.10	3.54 ± 0.11
Specificity-controlled WD 6	2.81 ± 0.09	2.91 ± 0.10	3.34 ± 0.09	2.31 ± 0.11	2.53 ± 0.12	2.93 ± 0.12	3.09 ± 0.10	3.41 ± 0.12
Specificity-controlled WD 8	2.70 ± 0.11	2.39 ± 0.12	2.54 ± 0.12	1.80 ± 0.13	2.00 ± 0.10	2.49 ± 0.12	2.47 ± 0.10	2.87 ± 0.11
Response-related control (WD)								
Response-related controlled WD -10	2.77 ± 0.12	2.45 ± 0.12	3.26 ± 0.11	1.96 ± 0.10	2.31 ± 0.12	2.47 ± 0.12	2.73 ± 0.11	3.12 ± 0.12
Response-related controlled WD 0	**2.87 ± 0.12**	2.97 ± 0.11	3.55 ± 0.09	**2.62 ± 0.11**	2.48 ± 0.10	2.88 ± 0.12	3.21 ± 0.09	**3.70 ± 0.10**
Response-related controlled WD 5	2.79 ± 0.10	2.83 ± 0.09	3.35 ± 0.12	2.40 ± 0.12	2.51 ± 0.13	2.80 ± 0.13	3.13 ± 0.12	3.41 ± 0.12
Response-related controlled WD 10	2.74 ± 0.11	2.42 ± 0.12	2.93 ± 0.11	1.95 ± 0.12	2.20 ± 0.12	2.56 ± 0.12	2.90 ± 0.12	3.12 ± 0.10
Response-related controlled WD 13	2.63 ± 0.12	2.06 ± 0.11	2.40 ± 0.09	1.74 ± 0.11	2.07 ± 0.11	2.25 ± 0.12	2.49 ± 0.14	2.63 ± 0.10

Table 4.8: Calibrated scores (mean ± std.) for all models and human evaluation metrics.

The first eight columns are Likert metrics on a 1–4 scale (except Avoiding Repetition, which is a 1–3 scale), where higher is better (except Inquisitiveness, which has an optimal score of 3). The last column, Persona Retrieval, is on a scale from 0 to 1 where higher is better.

The maximum of each column (excluding Human row) is in bold.

Rows marked with * are the six models included in Figures 4.5 and 4.6.

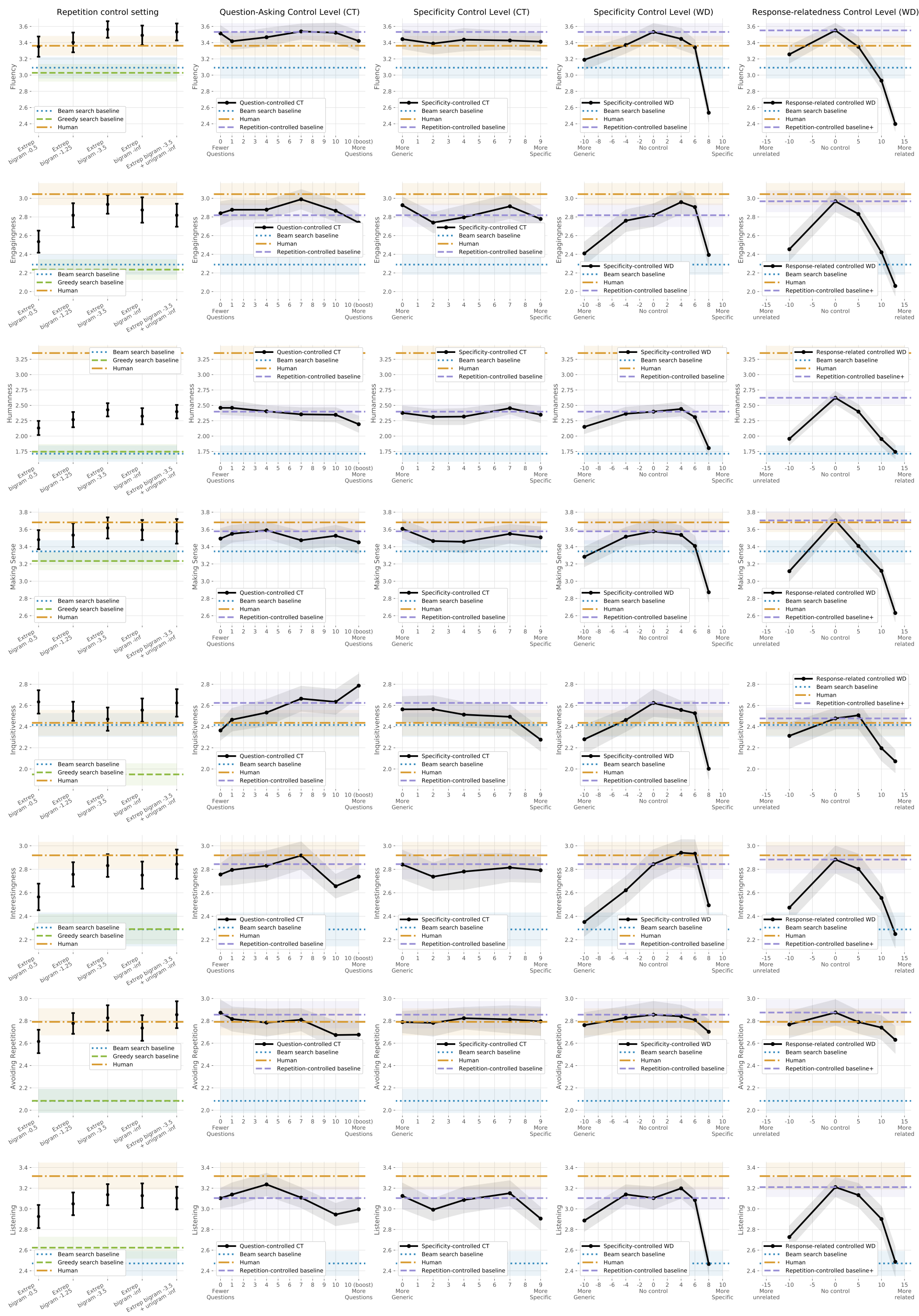


Figure 4.11: Calibrated human evaluation scores for all models. This is the same data as in Table 4.8. Note: ‘Repetition-controlled baseline+’ in the rightmost column is ‘Response-related controlled WD 0’ in Table 4.8. See Table 4.5 for explanation.

4.12 Conclusion

What makes a good conversation? Through our evaluation, we showed that a good conversation is about balance – controlling for the right level of repetition, specificity and question-asking is important for overall quality. We also found that conversational aspects such as interestingness, listening, and inquisitiveness are all important – though optimizing these can introduce a trade-off against certain types of errors (such as repetitive, disfluent, or nonsensical output). Secondly, multi-turn evaluation is essential to study what makes a good conversation – multiple turns are required to reveal issues such as repetition, consistency, and question-asking frequency. Lastly, what do we mean by 'good'? Although humanness and engagingness are both commonly used as overall quality metrics, the two are very different. While our models achieved close-to-human scores on engagingness, they failed to get close on humanness – showing that a chatbot need not be human-like to be enjoyable. This striking result also demonstrates the importance of measuring more than one quality metric when evaluating dialogue agents. The evaluation difficulties I encountered in this work – the unnaturalness of the task, and the low-bar of crowdworker-measured human performance – motivated my work on real-user conversations in the Alexa Prize (Chapter 6).

Controlling generation. This chapter shows that neural generative systems have systemic problems when applied to open-ended dialogue, some of which (e.g., repetition) are only observable in the multi-turn setting. Furthermore, we showed that control of low-level attributes offers one practical way to correct these problems, yielding large improvements to overall quality – in our case, comparable to systems trained on much more data. Of the two control methods tested in this chapter, conditional training is more stable and principled than weighted decoding. Indeed, since the publication of this work, conditional training is more widely-used; for example Keskar et al. (2019) do large scale conditional pretraining. More generally, there is a trend towards including meta-data about the task to be performed in the textual input; see Section 2.4.

Large pretrained neural generative dialogue. Since this work, the most noticeable performance improvements in neural generative dialogue models have been achieved by larger pretrained models like DialoGPT (Zhang et al., 2020b), Meena (Adiwardana et al.,

2020) and BlenderBot (Roller et al., 2021). These models are trained on larger, more diverse datasets than PersonaChat – usually derived from Reddit. DialoGPT and Meena both report a type of human-level performance for their best models; in both cases this is measured with respect to crowdworker conversationalists.

Decoding techniques remain varied. Both DialoGPT and Meena use sample-and-rank: that is, use top-k (or natural sampling with a lowered temperature) to sample several candidates, then rerank them by the model's probability (Meena) or for mutual information via a backwards model's probability (DialoGPT). They find that sample-and-rank produces more interesting and diverse utterances that are preferred to greedy or beam search in human evaluation. Conversely, BlenderBot uses beam search (a comparison to top-k and sampling-and-rank is inconclusive). However, they they emphasize that for beam search, choosing the right beam size, controlling output length, and repetition blocking (similarly to this chapter) is very important.

Both BlenderBot and Meena note that cross-turn repetition (`extrep` in this chapter) and parroting the partner (`partnerrep`) are still problems, along with self-contradiction and commonsense. In general, it seems that larger models offer a better tradeoff between sensibleness (or *making sense*, in this chapter) and specificity – in fact, Adiwardana et al. argue that achieving lower perplexity is the key to improving the tradeoff.

Chapter 5

The Effect of Pretraining for Story Generation

5.1 Introduction

As we saw in Chapter 3, neural generation of longer, multi-sentence pieces of text is difficult, even when provided with detailed source material. At the time of the work in this chapter, pretrained generative language models were beginning to show the ability to generate much higher-quality samples of long, open-ended text (Section 1.3.2). In this chapter, we aim to understand how pretraining affects open-ended generation, while also focusing on the vital role of the decoding algorithm (Problem 1, Section 1.3.3). For our investigation, we choose open-ended generation in its purest form – unconstrained narrative generation, given a loose prompt.

In 2018, large-scale neural models such as ELMo (Peters et al., 2018), BERT (Devlin et al., 2019) and OpenAI GPT (Radford et al., 2018) emerged as a dominant approach in NLP. By pretraining on massive amounts of unlabeled text (often orders of magnitude larger than the the target task's labeled dataset), these models achieve state-of-the-art performance across a variety of Natural Language Understanding benchmarks. In particular, the OpenAI GPT2 language model (Radford et al., 2019) achieved state-of-the-art performance on several language modeling benchmarks, even in a zero-shot setting. While GPT2's performance as a language model is undeniable, its performance as a text generator is much less clear. Though the model has generated certain impressive samples of text – such as a widely-circulated passage about Ovid's Unicorn (Radford et al., 2019) – there has been no detailed study to formalize these observations.

In this chapter, we perform an in-depth study of the properties of text generated by GPT2-117 (aka GPT2-small; the smallest version of GPT2) in the context of story generation. By comparing to a state-of-the-art, specialized-architecture neural story generation model (Fan et al., 2018b), we ask the following questions. In what ways does a large amount of open-domain pretraining data change the characteristics of generated text? In what ways does it make no difference? And is a task-specific architecture necessary?

As discussed in Section 1.3.3, for any probabilistic language model, the generated text is strongly affected by the choice of decoding algorithm – this is especially true for *open-ended* text generation tasks such as storytelling and chitchat dialogue (Kulikov et al., 2019; Holtzman et al., 2020). Nevertheless, most natural language generation papers evaluate only *one* decoding algorithm – this is often due to the time and expense required for human evaluation. For example, Fan et al. use top-k sampling (a decoding algorithm in which k governs the quality-diversity tradeoff; see Section 2.6.2), but only evaluate one value of k. However, evaluating one k gives an incomplete view of the generation system – several researchers have emphasized the importance of evaluating generation systems over the entire quality-diversity spectrum, rather than a single point on it (Caccia et al., 2018; Hashimoto et al., 2019).

In this study, we prioritize evaluating text across the *whole* k spectrum, and measuring *many* different automatic metrics, rather than a few human metrics. Though the lack of human evaluation limits our ability to measure overall quality (Liu et al., 2016; Novikova et al., 2017; Hashimoto et al., 2019), we are able to produce an objectively defined, richly detailed and reproducible evaluation of the generated text. To our knowledge, this work was the first comprehensive analysis of the characteristics of GPT2-generated text. Our study provides insight into the effect of large-scale pretraining on open-ended natural language generation, as well as the effect of k on text generated with top-k sampling. We hope our results will inform other researchers' choice of models, pretraining schemes, and decoding algorithms – decisions that can often feel like blind choices. To enable readers to browse the generated text, conduct their own evaluations, or run our evaluations on their own text, we

Prompt: You just discovered England's biggest secret: every time they say long live the queen it extends the queens life.	
Human-Written Story: I am immortal, I have inside me blood of kings. My name in Charles Windsor and I am heir to a throne. Prince of Wales, they've called me that my entire life and now I lay on my death bed and my mother - still going strong. I'm not sure how it's possible but she seems ageless, almost timeless. My whole life I've waited for that [...]	
Fusion Model ($k = 10$): *I have no idea when my father would come to visit this place, I was always so excited. I was always so scared. I had to go back to work. I was a little scared of him, but when I was younger, he had never come in for a visit. His mother told me, "I'm so scared", as he always did. She never talked to me about anything* [...]	**GPT2-117** ($k = 10$): ** " So what are we going to do? " * It was the first time she asked me this question that I ever had asked her. It has taken years but finally I am finally able to do it. It seems that this queen is the queen of England and she's finally the one to tell me that this queen is the reason why she's not being allowed to die.* [...]
Fusion Model ($k = 1000$): *"Where am I? What happened?" "Having been working on my job for over 6 hours now, I do not know how you worked!" "I have been working for the last three years. Surely I am an ancient god now." The bar patrons snickered. "Hello?" "Those last three years have been worse than a year ago." Pain.* [...]	**GPT2-117** ($k = 1000$): *It was an odd occasion for the Queen of England to meet with her. The plane sat idle at 3:26 PM on a Thursday night. Yesterday, the Queen had taken it upon herself to try and get a good look at the plane which had recently been found abandoned. A copious amount of curious glances from around the room until* [...]

Table 5.1: A prompt and human story from the dataset, plus the models' top-k generated stories, for two values of k.

publicly release our generated stories and evaluation code.[1]

5.2 Background

WritingPrompts dataset WritingPrompts (Fan et al., 2018b) is a story generation dataset containing 303,358 human-written (*prompt*, *story*) pairs collected from the /r/WritingPrompts subreddit – a forum where Reddit users compose short stories inspired by other users' prompts. An example can be seen at the top of Table 5.1. The mean prompt length is 28.4 words and the mean story length is 734.5 words. The dataset is 887MB of text in total, contains 200 million story words, and is divided into 90% train, 5% validation and 5% test splits.

[1] Code and generated stories at https://github.com/abisee/story-generation-eval

The Fusion Model The Fusion Model is a state-of-the-art neural story generation architecture trained on the WritingPrompts dataset (Fan et al., 2018b). It is based on the Convolutional Seq2seq model of Gehring et al. (2017) and aims to improve two aspects of story generation: modeling long-range context and increasing relevance of the story to the prompt. To achieve the former, the model uses a multi-scale gated self-attention mechanism. For the latter, the model uses a fusion mechanism (Sriram et al., 2018) in which one seq2seq model is trained on the task, then frozen, and a second seq2seq model is trained on the task with access to the first model's hidden states. Compared to the Convolutional Seq2seq model and other baselines, the Fusion Model achieves improved perplexity, story-prompt relevance and human preference scores. The Fusion Model has a vocabulary of 104,960 words, a 3-layer encoder and 8-layer decoder in the first seq2seq model, and a 5-layer encoder and 5-layer decoder in the second model – in total, 255.4 million parameters. Fan et al. use top-k sampling (Section 2.6.2) with $k = 10$ to generate stories.

GPT2-117 GPT2 (Radford et al., 2019) is a large Transformer language model trained on WebText, a diverse corpus of internet text (not publicly released) containing over 8 million documents equalling 40GB of text in total. The full-size GPT2 model, which has 1542 million parameters, obtains state-of-the-art results on a variety of language modeling and other Natural Language Understanding benchmarks. At the time of our experiments, Radford et al. had only released the smallest of the models, known as GPT2-117.[2] This model, which we use for our experiments, has 12 layers and 117 million parameters. Like the full-size GPT2 model, it has a vocabulary of 50,257 byte-pair-encoding (BPE) tokens (see Section 2.4). The model has a context size of 1024, meaning it can process text up to 1024 BPE tokens in length. Radford et al. show impressive samples of generated text, primarily from the full-size GPT2 model, generated with top-k sampling (Section 2.6.2) with $k = 40$.

[2]Since publishing this work, the larger GPT2 models have been publicly released.

5.3 Experimental Details

Preprocessing Fan et al. truncate WritingPrompts stories to 1000 words before training and testing. Due to the limited context size of GPT2-117, we additionally exclude (*prompt*, *story*) examples that are longer than 1024 BPE tokens when concatenated. The resulting dataset, which we call WritingPrompts-1024, has 192,364 training, 11,115 validation, and 10,686 test examples.

The Fusion Model We use the pretrained version of the Fusion Model, which is available in the Fairseq framework (Ott et al., 2019). For comparability with GPT2-117, we evaluate the Fusion Model on WritingPrompts-1024 (see Table 5.2), obtaining perplexities similar to those reported by Fan et al. on the full WritingPrompts dataset.

GPT2-117 In order for the model to condition on prompts and generate stylistically correct stories, we finetune GPT2-117 on WritingPrompts-1024.[3] We frame WritingPrompts as a language modeling task, representing the prompt and story as a single sequence separated by a delimiter token. We finetune the pretrained model until convergence using the default hyperparameters provided in the HuggingFace repository (though we reduce batch size to fit on a single GPU), and use the finetuned model for all further evaluations.

We compute the *word-level* perplexity of the finetuned GPT2-117 on the WritingPrompts-1024 dataset. That is, we normalize the total negative log probability of the target text by the number of *word*-level (i.e., Fusion Model) tokens, not the number of BPE tokens. This enables us to compare the perplexities of the two models, despite the tokenization difference (Radford et al., 2019). The finetuned GPT2-117 obtains a test set word-perplexity of 31.54[4] – six points lower than the Fusion Model.

Generation settings For both models, we generate stories using top-k sampling, obtaining 1000 stories (from 1000 different test set prompts) for several values of k ranging from 1

[3] We use the PyTorch re-implementation of GPT2-117 available at `https://github.com/huggingface/pytorch-transformers`

[4] This is similar to other GPT2-117 WritingPrompts finetuning experiments (Mao et al., 2019; Ziegler et al., 2019).

Model	**Valid ppl**	**Test ppl**
Fusion Model	37.05	37.54
GPT2-117	31.13	31.54

Table 5.2: Word-level perplexities on WritingPrompts-1024 for the Fusion Model and finetuned GPT2-117.

to vocabulary size. We use softmax temperature 1. Like Fan et al., we generate exactly 150-word stories and block the Fusion Model from generating `<UNK>`.

To obtain human-written stories for comparison, we truncate WritingPrompts-1024 test set stories to 150 words (discarding those shorter than 150 words). To reduce variance, measurements for human stories are computed over this entire set (rather than just 1000 stories).

5.4 Story-prompt Relatedness

Prior research has observed that seq2seq systems frequently produce text that is unrelated to the provided context – particularly under likelihood-maximizing decoding algorithms such as beam search. The issue has inspired multiple explanations (Jiang and de Rijke, 2018) and multiple solutions – such as alternative training objectives (Li et al., 2016a), decoding objectives Baheti et al. (2018), and architectural changes (Fan et al., 2018b). This issue was also explored in Chapter 4. In this section, we measure how strongly the models condition on the prompt.

Prompt ranking accuracy For both models, we compute *prompt ranking accuracy* (Fan et al., 2018b), which measures the language model's sensitivity to the provided prompt. Following the methodology of Fan et al., we randomly select 1000 human-written stories from the test set, and measure the probability (according to the model) of each story conditioned on 10 different prompts – the true prompt, plus nine randomly selected prompts. The prompt ranking accuracy of a model is the percentage of cases in which the model assigns a higher probability to the story under its true prompt than under all of the other nine.

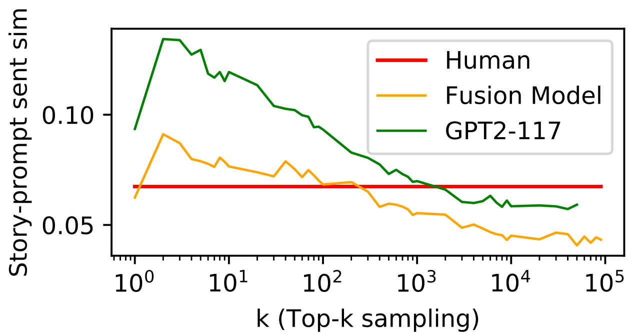


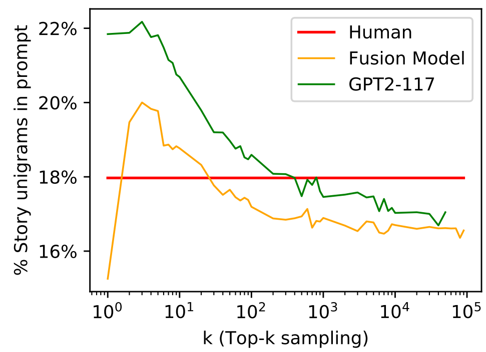
Figure 5.1: Compared to the Fusion Model, GPT2-117 produces stories that are more semantically similar to the prompt. Similarity decreases as k increases.

We find that GPT2-117 scores **80.16%** on this task, while the Fusion Model scores **39.8%**.[5] Random chance scores 10%. This striking result indicates that GPT2-117 conditions on the prompt much more strongly than the Fusion Model. This is notable, especially because the fusion technique is intended to improve story-prompt relevance.

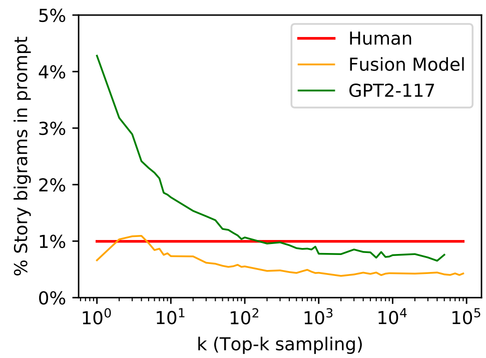
N-gram similarity For $n = 1, 2, 3$, we measure the percentage of generated n-grams that also appear in the prompt. For all n and k, we find that GPT2-117 has a higher overlap (i.e., copies more from the prompt) than the Fusion Model – see Figure 5.2. Furthermore, for $k < 100$, the GPT2-117 overlap is generally much higher than human levels. Both these phenomena can be seen in Table 5.1, where, for $k = 10$, GPT2-117 copies words such as *queen* more often than both the Fusion Model and the human-written story.

Sentence embedding similarity To capture a higher-level notion of semantic similarity, we measure *story-prompt sentence similarity* – the cosine similarity of story-prompt sentence pairs, averaged by taking the mean over all pairs. Sentences are represented by the embedding method of Arora et al. (2017) – a weighted average of the GloVe embeddings (Pennington et al., 2014) of the words, with the first principal component removed. As shown in Figure 5.1, we find a similar pattern as for n-gram similarity: GPT2-117 generates

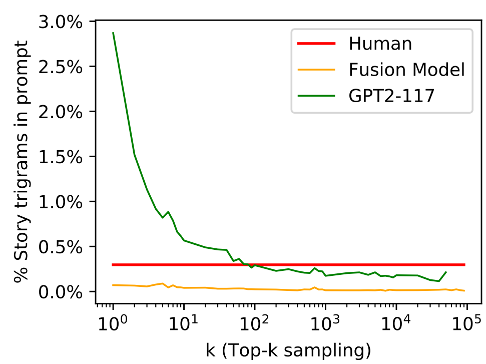
[5] Fan et al. (2018b) report a prompt ranking accuracy of 16.3% for the Fusion Model. We provided the authors with our prompt ranking accuracy code (which was built on top of the authors' code). The authors indicated that the discrepancy may be due to some code version changes between the time of their original experiments and their code release.



(a) Percent of all story unigrams that are in the prompt.

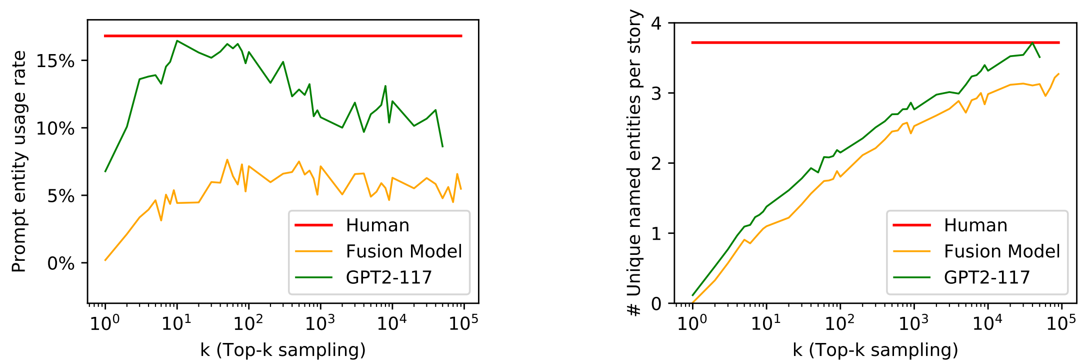


(b) Percent of all story bigrams that are in the prompt.



(c) Percent of all story trigrams that are in the prompt.

Figure 5.2: n-gram similarity between prompt and story , for $n = 1, 2, 3$, for both models and all k. GPT2-117 copies many more n-grams from the prompt than the Fusion Model.



(a) The proportion of all prompt named entities that are used in the story.

(b) The number of unique named entities that appear in the story.

Figure 5.3: Prompt entity usage rate (left) and mean number of unique named entities in the story (right), for both models and all k. GPT2-117 generally uses a larger proportion of the prompt named entities, and more named entities overall, than the Fusion Model. Both models generally use fewer named entities than human text when k is less than vocabulary size.

sentences that are more similar to the prompt than the Fusion Model for all k, and both models' prompt similarity decreases as k increases.

Named entity usage Generally, most named entities mentioned in the prompt (such as *Queen* and *England* in Table 5.1), should also be mentioned in the story. Using the spaCy named entity recognizer,[6] we measure the *prompt entity usage rate*, which is the percentage of all prompt named entities that appear in the story.[7] As shown in Figure 5.3, we find that GPT2-117 uses more of the prompt named entities than the Fusion Model (as well as more named entities overall), but both models use fewer named entities than humans when k is less than vocabulary size.

These patterns can be seen in Table 5.1: GPT2-117 uses the prompt entities *Queen* and *England* whereas the Fusion Model does not (for either k), and GPT2-117 uses specific time entities such as *Thursday* and *3:26 PM*. While the human story introduces highly-related

[6] https://spacy.io

[7] Given that we limit stories to 150 words, this percentage is lower than it would be if we generated longer stories.

entities such as *Charles Windsor* and *Prince of Wales* that were not in the prompt, neither model does this (for either k).

Conclusion In this section, we found that GPT2-117 conditions on the prompt much more strongly than the Fusion Model – a result which holds both in language modeling and generation settings. The latter result supports Radford et al.'s informal observation that GPT2 has a 'chameleon-like' ability to 'adapt to the style and content of the conditioning text'.[8] We speculate that GPT2-117's stronger conditioning ability may derive from its Transformer decoder architecture, whose powerful self-attention is used for story-prompt attention. Though the Fusion Model uses a similar self-attention mechanism in the decoder (i.e., story side), the prompt-story attention has a simpler formulation – for example, there are no separate key and value vectors (Gehring et al., 2017). Lastly, we note that very strong prompt-conditioning is not always a good thing – GPT2-117 often generates stories that copy too much or too literally from the prompt when k is small (this can be seen in Figure 5.2).

5.5 Coherence

A good story generation model should produce coherent text with a logical ordering of events. Similarly, the underlying language model should be a good coherence scorer – assigning higher probability to coherent text than incoherent text. Barzilay and Lapata (2008) evaluate a coherence scorer by measuring its ability to rank shuffled human-written text as less coherent than the original unshuffled text. We use this method to evaluate our story generation models.

For each story in the test set, we select the first 15 sentences. We then produce 14 corrupted versions of the story by switching each pair of adjacent sentences. We use the language model to compute the probability of each of the 14 corrupted stories, as well as the original story. The model's error rate is the percentage of cases in which it rates any of the 14 corrupted candidates better than the original candidate. Random guessing yields 93.33% error. Both models perform well on this task – the Fusion Model has an error rate

[8] https://openai.com/blog/better-language-models/

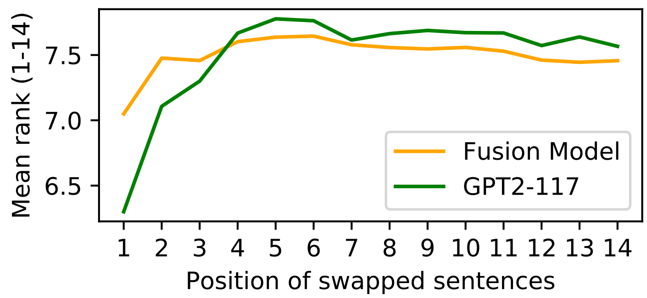


Figure 5.4: Sensitivity of the models to swapped sentences in different positions. A higher mean rank indicates higher sensitivity (i.e., the model assigns lower probability) relative to other positions. Both models are less sensitive to swapped sentences at the beginning of the text, compared to later. GPT2-117 shows this pattern more strongly, indicating greater use of context.

of **3.44%** and GPT2-117 an error rate of **2.17%**. This 36.92% error reduction indicates that GPT2-117 is more sensitive to ordering of events.

We also investigate how the position of the swap affects its plausibility (relative to other positions). Figure 5.4 shows, for each swap position, the mean rank assigned to that swap by the model (where rank 1 is the most probable of the 14 corrupted candidates, and rank 14 the least probable). GPT2-117 assigns a much lower rank to the first few swap positions (i.e., rates them more probable) than the later positions. The Fusion Model shows a similar but less pronounced pattern. This shows that both models are less sensitive to out-of-order sentences that occur at the beginning of the text, than those occurring later.[9] The stronger pattern for GPT2-117 may be due to its stronger context conditioning (as shown in Section 5.4) – thus becoming more sensitive as context increases. However, even for the first three swaps, GPT2-117 is more accurate than the Fusion Model at distinguishing the swapped text from the original.

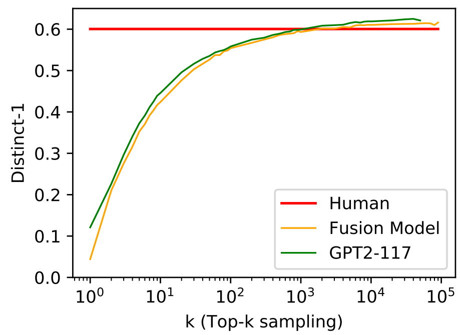
[9]It's also possible that out-of-order sentences are inherently harder to detect at the beginning of text.

5.6 Repetition and Rareness

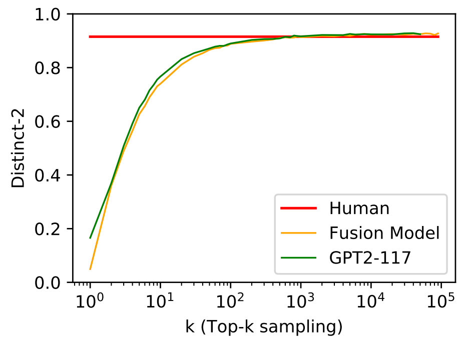
Generic, under-diverse and repetitive text is a well-documented problem in neural text generation (Jiang and de Rijke, 2018). While there are many proposed solutions to the problem (Li and Jurafsky, 2016; Vijayakumar et al., 2018; Baheti et al., 2018; Zhang et al., 2018a), including in Chapter 4, it has been shown that a primary cause is likelihood-maximizing decoding algorithms such as greedy decoding and beam search (Holtzman et al., 2020). In this section we investigate the role of large-scale pretraining, and the role of k, in this problem.

N-gram repetition The *distinct*-n metric of a piece of text is the number of unique n-grams divided by the total number of generated n-grams (Li et al., 2016a). We measure distinct-n of the generated stories for $n = 1, 2, 3$. A high ratio indicates a high level of within-story lexical diversity, while a low ratio indicates a large amount of within-story repetition. As shown in Figure 5.5a, both models' unigram diversity is far below that of human text when k is small. For example, at $k = 10$ (the setting used by Fan et al.), the Fusion Model obtains a distinct-1 of 42.4%; much less than the human level of 60.0%. This results in a high level of repetition, as shown in Table 5.1: for $k = 10$, both models repeat many phrases (such as *always*, *so scared*, and *finally*).

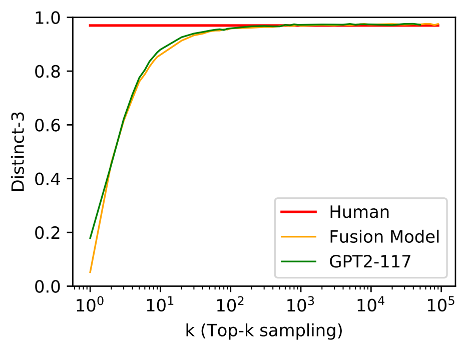
For bigrams and trigrams, the pattern is similar to unigrams (see Figure 5.5). For both models, distinct-n increases as k increases, converging to a value close to the human level as k approaches vocabulary size. Though GPT2-117 has a slightly higher distinct-n than the Fusion Model for most values of k, the difference is negligible compared to the influence of k. We make three conclusions from these patterns: (1) Our findings support Holtzman et al.'s observation that repetition is strongly related to choice of decoding algorithm, and that likelihood-maximizing algorithms (such as top-k sampling with low k) are a primary cause of repetition. (2) The models have in fact learned the correct rate of repetition in human text – they are able to match this rate when they sample from their full (untruncated) distribution. (3) Repetition is unlikely to be solved by more pretraining data alone – even though GPT2-117 is trained on 45 times as much data as the Fusion Model, it produces text that is almost equally repetitive (for equal k).



(a) Distinct-1 (ratio of unique unigrams in the story to total # generated unigrams in the story).

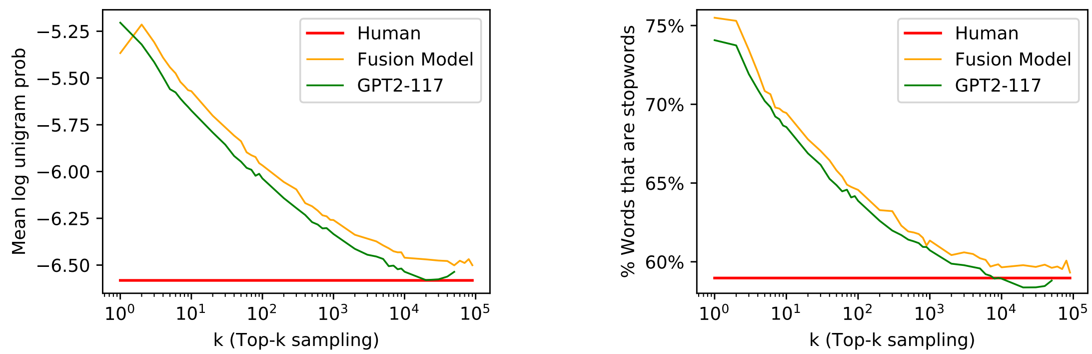


(b) Distinct-2 (ratio of unique bigrams in the story to total # generated bigrams in the story).



(c) Distinct-3 (ratio of unique trigrams in the story to total # generated trigrams in the story).

Figure 5.5: Distinct-n for $n = 1, 2, 3$, for both models and all k. Repetition (low distinct-n) is primarily caused by choice of decoding algorithm (here low k), not insufficient training data. GPT2-117 is trained on $45\times$ more data than the Fusion Model, but is similarly repetitive for all k.



(a) The mean log unigram probability of generated words. Higher values indicate using fewer rare words while lower values indicate using more rare words.

(b) The percent of generated words that are stopwords, for both models, across different k. We use the NLTK English stopword list.

Figure 5.6: Rare word usage metrics for both models and all k. GPT2-117 produces slightly more rare words (left) and slightly fewer stopwords (right) than the Fusion Model, for equal values of k. These rareness metrics do not reach human levels until k is close to vocabulary size.

Rare word usage We compute the *mean log unigram probability* of the words in the generated story[10] – a high value indicates using fewer rare words while a low value indicates more rare words. As shown in Figure 5.6, word rareness is primarily governed by k – however, GPT2-117 has a lower mean log unigram probability (i.e., uses more rare words) than the Fusion Model for all equal values of $k \geq 2$. This can be seen for example in Table 5.1, where GPT2-117 generates rarer words such as *idle* and *copious* for $k = 1000$. GPT2-117 also generates fewer stopwords than the Fusion Model, for all equal k.

GPT2-117's slightly higher rare word usage (compared to the Fusion Model) might be explained by: (1) its BPE encoding, which allows it to generate new words, not just those in a fixed vocabulary; (2) pretraining on a large amount of diverse text, allowing it to learn to produce a greater variety of words; (3) stronger conditioning on the prompt as described in Section 5.4 – which may inject more rareness into the generated text.

[10]The unigram probability distribution was calculated with respect to the WritingPrompts training set.

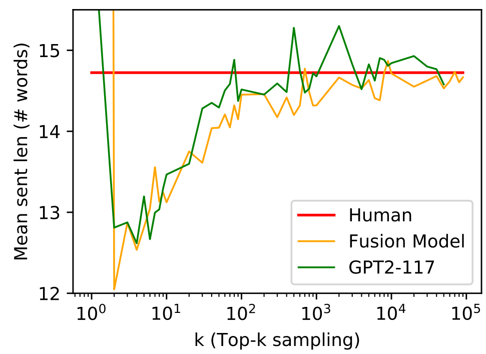


Figure 5.7: Mean sentence length for both models and all k. For both models, sentence length increases as k increases. The spike at $k = 1$ is due to long repeating sequences with no sentence-ending token.

Conclusion Choice of decoding algorithm is a primary factor in diversity and repetition problems, with likelihood-maximizing algorithms (such as top-k with small k) the main culprit. Although GPT2-117 generates more rare words and is very slightly less repetitive than the Fusion Model, the difference is small compared to the effect of k. This indicates that solving repetition and diversity (Problem 2, Section 1.3.3) will most likely require changes to the decoding algorithm, and not just scaling up the training data.

5.7 Syntactic Style and Complexity

A well-trained story generation model should match both the syntactic style and complexity of its training data. Low complexity can be a sign of less sophisticated writing, while high complexity can be a sign of poor readability (Beers and Nagy, 2009; McNamara et al., 2010). In this section, we measure some features related to the syntactic style and complexity of the generated stories.

Sentence length Sentence length is a simple but effective feature to estimate readability and syntactic complexity of text (Kincaid et al., 1975; Roemmele et al., 2017). We find that both models generate sentences that are on average shorter than human sentences when k is small, but converge to approximately human length as k increases (see Figure 5.7).

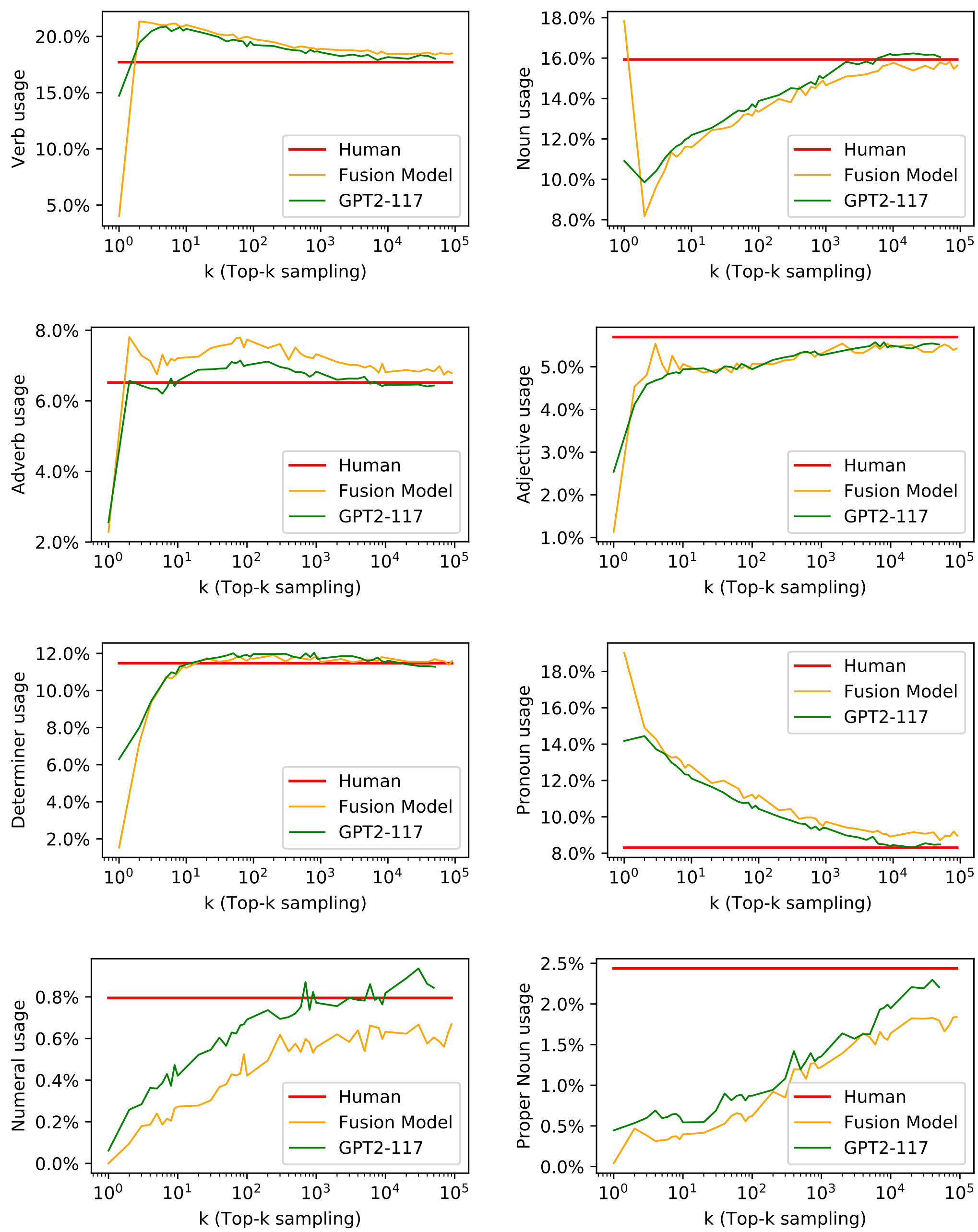
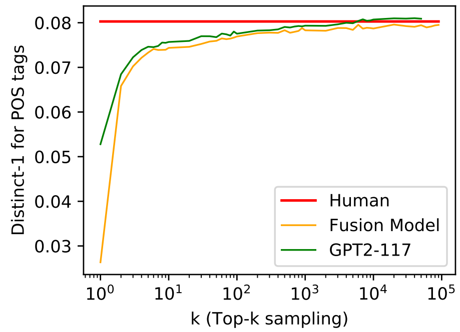
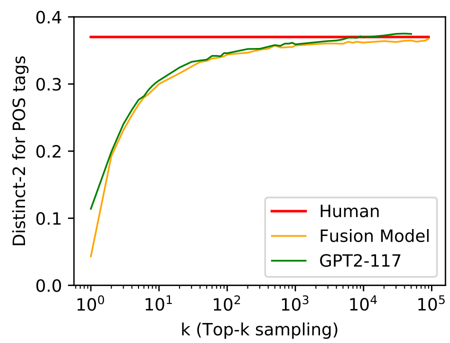


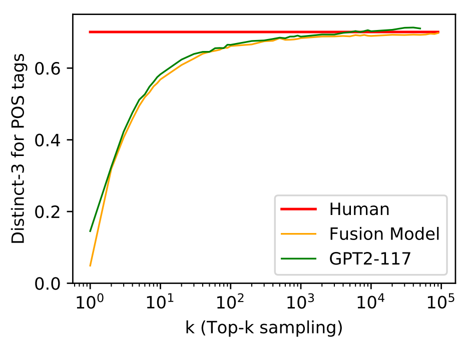
Figure 5.8: Usage of different POS tags in the generated stories. GPT2-117 tends to fit the human distribution more closely than the Fusion Model as k approaches vocabulary size, in particular producing more specific POS categories such as Numeral and Proper Noun. When k is small, generated text is characterized by more verbs and pronouns, and fewer nouns, adjectives, numerals and proper nouns, than human text.



(a) POS tag distinct-1 (ratio of unique POS unigrams to total # generated POS unigrams in the story).



(b) POS tag distinct-2 (ratio of unique POS bigrams to total # generated POS bigrams in the story).



(c) POS tag distinct-3 (ratio of unique POS trigrams to total # generated POS trigrams in the story).

Figure 5.9: POS tag distinct-n for $n = 1, 2, 3$, both models and all k. Distinct-n, which represents syntactic diversity, increases with k. GPT2-117 reaches human levels at $k = 6000$ for unigrams, $k = 9000$ for bigrams, and $k = 6000$ for trigrams. Syntactic diversity is slightly higher for GPT2-117 than Fusion Model for equal k, but the primary factor is k.

Part-of-speech usage It has been shown that the distribution of parts-of-speech (POS), and more generally the distribution of POS n-grams[11] is a useful feature to represent the style of a piece of text (Argamon et al., 1998; Ireland and Pennebaker, 2010; Roemmele et al., 2017).

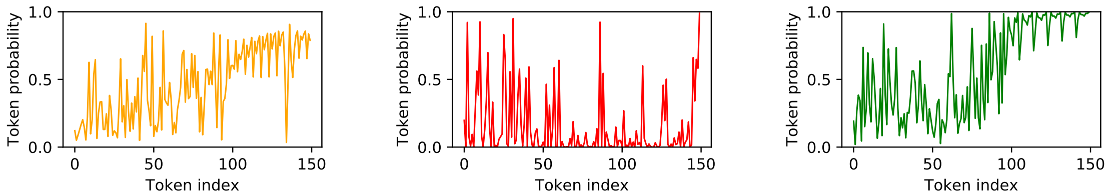
Firstly, we compare the part-of-speech distributions of the model-generated text and the human text (see Figure 5.8). Both models (especially GPT2-117) closely fit the human POS distribution as k approaches vocabulary size.[12] This implies that, as with lexical diversity, the models have no difficulty fitting the statistical distribution of human syntax. However, under likelihood-maximizing decoding algorithms such as low k, a completely different distribution emerges, in which text contains more verbs and pronouns than human text, and fewer nouns, adjectives and proper nouns.

Secondly, we measure the syntactic diversity of the text using the distinct-n metric for POS n-grams ($n = 1, 2, 3$) – see Figure 5.9. As with lexical diversity (see Section 5.6), we find that syntactic diversity is similar for the two models, is very low when k is small, and matches human level as k approaches vocabulary size. It's likely that for low k, the syntactic under-diversity of the text is largely caused by lexical under-diversity (i.e., repetition). However, we note that as k increases, lexical diversity reaches human level sooner than syntactic diversity – for example, GPT2-117's lexical distinct-3 reaches human level at $k = 600$ (Figure 5.5c), but its POS distinct-3 reaches human level at $k = 6000$ (Figure 5.9c). This implies that, even when the text is no more repetitive than human text, it may still be syntactically repetitive (using the same part-of-speech patterns repeatedly).

Conclusion We find when k is small, syntactic complexity of generated text is low, consisting of shorter sentences and a narrower range of syntactic patterns. However, as k approaches vocabulary size, the syntactic style of generated text closely matches human syntactic patterns. As with n-gram diversity in Section 5.6, our results show that syntactic under-diversity is primarily caused by low k, not insufficient training data.

[11] For example, the sentence *I like cats* has the POS bigrams PRONOUN VERB and VERB NOUN.

[12] One exception is Proper Noun: both models fail to produce enough of these even as k approaches vocabulary size.



(a) **Fusion Model** ($k = 2$): *I had never seen a man so young before. I had never seen him before, but he had always seemed to be a man of a man. He was young, and he was young. He was a man of a man, and a man who was young, and a man who was* [...]

(b) **Human Text**: *"Looks like the rain's stopped." I peered out the window. Art was right; time to get to work. "Alright, let's move out." I could hear the scraping of the stone armor as the men slowly stood. Despite the training,* [...]

(c) **GPT2-117** ($k = 2$): *I've always been a man of the people. I've always been a strong man. I've always been a strong man. I was born in the city, I was raised in the country. I was raised in a family that wasn't very good. I 'm not a good man.* [...]

Figure 5.10: Under top-k sampling with small k ($k = 2$), the two models (left and right) produce text that falls into increasingly confident repeating loops. By contrast, human text (center) maintains an irregular pattern of surprising (low probability) tokens. The human text probabilities are measured with respect to the Fusion Model, but similar patterns hold for GPT2-117. Inspired by Holtzman et al. 2020's figure showing probabilities under beam search.

5.8 The Element of Surprise

Model confidence over time Several researchers have observed that *model over-confidence* (the model placing high probability on a small range of tokens) can cause poor quality generation (Jiang and de Rijke, 2018; Holtzman et al., 2020). In particular, they show that for likelihood-maximizing decoding algorithms such as beam search, model confidence can increase in a snowball-like effect, getting stuck in a loop of repetitive but increasingly self-confident text. We observe this problem in both our models when k is small. For example, in Figure 5.10, both models fall into self-reinforcing repetitive loops with rising confidence. The loop is difficult to break – the Fusion Model briefly escapes (shown as a sudden downwards spike), but quickly returns. By contrast, the human text does not show a

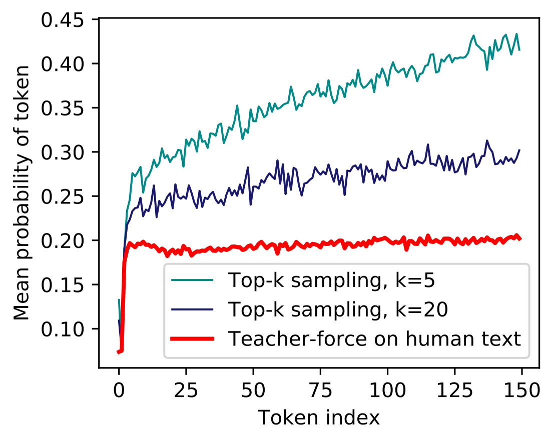


Figure 5.11: Mean probability for each of the first 150 word-level story tokens. When teacher-forcing the model on human text, probability increases slowly. When generating with top-k sampling, probability increases faster, especially for smaller k. This plot is for the Fusion Model; similar patterns hold for GPT2-117.

strong rising trend in probability, and intermittently uses low probability words throughout.[13]

We formalize these anecdotal observations by measuring the average probability of each of the first 150 word-level tokens in the story (Figure 5.11). We find that even when teacher-forcing on human text, the token probabilities increase slightly as the story progresses. This is likely due to the usefulness of additional context, which increases the model's prediction accuracy. By comparison, we find that when generating with top-k sampling, the probabilities increase more rapidly, and the increase is even more rapid for smaller k. This confirms that likelihood-maximizing decoding algorithms (such as top-k sampling with small k) lead to more rapidly increasing model over-confidence. Furthermore, we find this pattern holds for both models, with probabilities increasing at a similar rate for equal k. This indicates that, like repetition, model over-confidence is unlikely to be solved by more training data, and is largely governed by choice of k.

Overall model confidence We also measure the models' overall confidence, as represented by the total log probability (according to the model) of the generated story. For both

[13] Gehrmann et al. (2019) also identify presence of low probability words as an indicator of human-generated text.

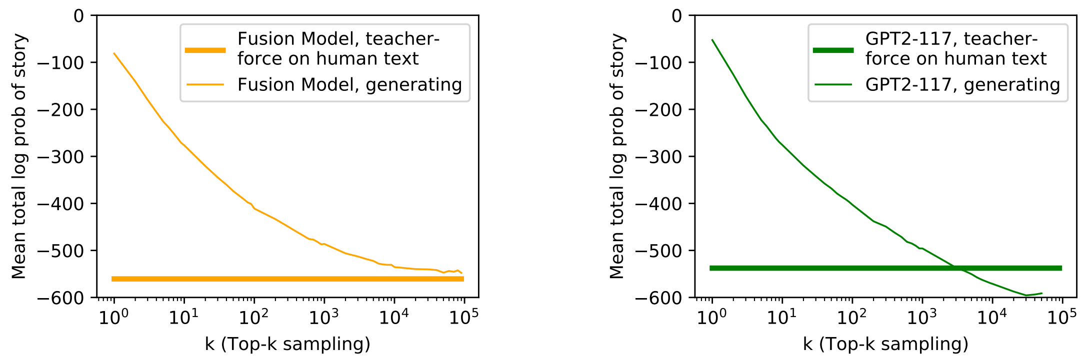
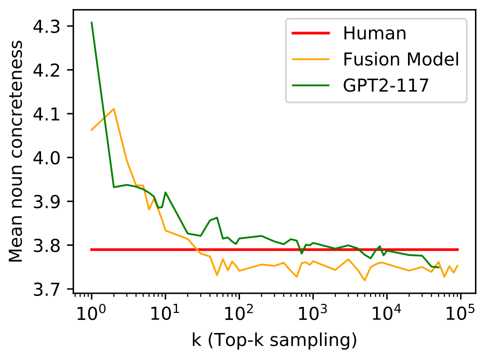


Figure 5.12: The mean total log probability of the story (150 words), as measured by the models on their own generated output and on human-written stories. Interestingly, the Fusion Model (left) converges to the same probability it assigns to human-written stories as k approaches vocabulary size, whereas GPT2-117 (right) converges to a lower probability.

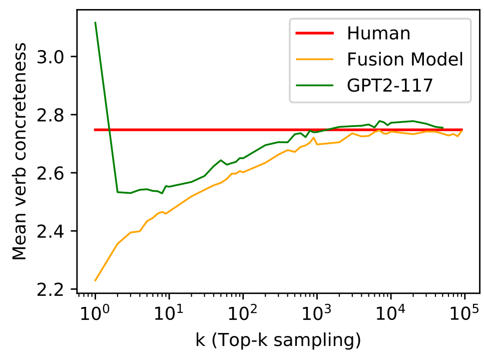
models, we find that story probability decreases as k increases – see Figure 5.12. This makes sense, as higher k means sampling tokens with lower probability. As k approaches the vocabulary size, the Fusion Model's generated story probability matches the probability it assigns to human-written WritingPrompts stories. Interestingly however, the same is not true for GPT2-117, which converges to a story probability that is *lower* than the probability it assigns the human stories. This means that under full (non-truncated) sampling, the Fusion Model produces text that is *equally surprising* (to itself) as the WritingPrompts stories, whereas GPT2-117 produces text that is *more surprising* to itself. Explaining this observation is an open question – we speculate that GPT2-117's WebText pretraining may cause it to generate (under high k) text in a style or genre that is less predictable than WritingPrompts stories.

5.9 Concreteness

Brysbaert et al. (2014) define the *concreteness* of a word as 'the degree to which the concept denoted by a word refers to a perceptible entity'. Concrete words are generally easier to remember than abstract words, and psycholinguists have theorized they may be learned differently (i.e., concrete words by direct experience and abstract words by text



(a) Mean concreteness rating (1-5) of nouns in the story.



(b) Mean concreteness rating (1-5) of verbs in the story.

Figure 5.13: Mean concreteness rating of the nouns and verbs in the story , for both models and all k. GPT2-117 generally produces nouns and verbs that are more concrete than the Fusion Model for the same k. For both models, as k increases, noun concreteness reduces and verb concreteness increases.

and discourse). Brysbaert et al. provide human concreteness ratings for 40,000 common English lemmas rated on a scale from 1 to 5.[14] We use these ratings to measure the mean concreteness of the nouns and verbs in the story text – see Figure 5.13.

We find that for the same k, GPT2-117 tends to generate more concrete words than the Fusion Model, and that for both, concreteness converges to approximately human levels as k increases. Interestingly, when k is small, the noun concreteness is much *higher* than human levels, whereas the verb concreteness is much *lower*. This indicates that for small k, both models produce stories that, compared to human-written stories, have too many physical objects (as opposed to abstract nouns), and too few physical actions (as opposed to abstract verbs). This reflects the trend demonstrated in Table 5.1: when k is small, the models tend to generate descriptive sentences with mostly *is* verbs (e.g., *I was always so excited*), and physical nouns (e.g., *mother, father, queen*). Only when k increases do we see more tangible actions (e.g., *The bar patrons snickered*) and abstract nouns (e.g., *pain, glances*). Table 5.3 shows a detailed example, with all nouns and verbs annotated with concreteness.

[14]For example, the nouns *television*, *darkness*, and *idea* are rated 4.83, 3.85 and 1.61 respectively, and the verbs *talk*, *see*, and *hope* are rated 4.07, 3.21 and 1.25 respectively.

Prompt: A man finally discovers his superpower... well into his 80's.

Fusion Model ($k = 10$):
I be a child . bear into a world of darkness , and in the world , there be only a few people . My mother and I be always alone . She have never be like this . But , she have no idea what we be together to be . We be always alone . We be always together . And , of course , she have never exist . But , I never think about it , because she be always alone . In the world , it was n't like she have a power . I be always alone , and there be a time when the light would turn on . There be a time when I could see the light , and I could see it in her eyes , and I could see the light , and I could see it

Mean noun concreteness: 3.892
Mean verb concreteness: 2.173

GPT2-117 ($k = 10$):
A man in an old white coat sit in a corner. He wear a black jacket with a blue tie. His hair be grey. His eyes be black, but his voice be deep and clear. He look to where the man be sit, and he see a smile on his lips. It be a smile he know he would see from his own eyes. But he be too late. He be on the sidewalk by the river when the man come. He be wear a black coat with a purple tie. He have a black tie and a white shirt. But he be still wear a white suit. And it seem he would look back at him. A smile on his face. A look his friend do n't recognize. He have no

Mean noun concreteness: 4.720
Mean verb concreteness: 2.488

Fusion Model ($k = 1000$):
For a brief moment, the dark green of his eyes flash the last lines of reality into existence, finally fade slowly into existence so that he could not feel it. Only the millions of pixels float in his peripheral vision almost disappear. His radio respond to the pinging very hard silence of the outside world. Seven people have visit his main internal network for what seem like a lifetime but this time, the only switch to an external supply system that he could simply take advantage of. Unable to convey feelings about the last words he would have to endure, but it have respond to the innumerable messages and countless sleepless hours. Most of them be always available on its surface, just to make sure. In his quest for to spend eternity on death, he send

Mean noun concreteness: 3.201
Mean verb concreteness: 2.435

GPT2-117 ($k = 1000$):
(First time poster, hope its ok) The young boy, watch tv, spot the television onscreen, before glance around to see the screen start the countdown on the tv, point to the screen in " It 's both the same. " "... let 's... let 's try this and... we will team up so that... we can... have the same power....like... so we can use this superpower over and over again. " A brief silence. Only a familiar conversation, interrupt his mad dash movement, follow with his high pitch slurred and wither voice : " I ca n't stand anyone talk like that son*s*. " More casual conversation that interrupt his childish step be rush to the scissors.

Mean noun concreteness: 3.793
Mean verb concreteness: 3.162

Table 5.3: Generated stories from both models, under $k = 10$ and $k = 1000$. Nouns are highlighted in green and verbs in yellow. The highlighting intensity reflects the word's concreteness rating. For equal k, GPT2-117 generally generates more concrete words than the Fusion Model. For both models, low k is characterized by high noun concreteness (e.g., physical objects such as *jacket*) and low verb concreteness (e.g., non-physical actions such as *be*). Conversely, high k is characterized by low noun concreteness (e.g., abstract concepts such as *reality*) and high verb concreteness (e.g., physical actions such as *talk*).

5.10 Conclusion

The effect of massive pretraining In this study, we find that GPT2-117 is arguably a better story generation model than the Fusion Model in several specific ways: it conditions much more strongly on the provided context, is more sensitive to correct ordering of events, and generates text that is more contentful (using more rare words, concrete words, and named entities). In particular, the stronger conditioning result is striking, as the Fusion Model is a complex task-specific architecture designed to increase story-prompt relevance. This demonstrates that sometimes, a general-purpose model architecture can outperform a complex task-specific architecture when provided with enough pretraining data.

However, we find that in other aspects, GPT2-117 performs *no better* than the Fusion Model: when k is small, the models generate text that is equally lexically under-diverse, syntactically under-complex, and repetitive – with a tendency to fall into a snowball effect of increasing over-confidence (Problem 2, Section 1.3.3). However, these problems correct themselves (i.e., the metrics match human levels) when the models generate from their untruncated distribution (aka natural sampling, Section 2.6.2). Our results show that these oft-cited neural generation problems are *not* the fault of the models themselves (which are in fact statistically well-trained to match human text for these metrics), nor caused by too little training data (as these problems are not improved by GPT2-117's extensive pretraining). Instead, they are primarily caused by likelihood-maximizing decoding algorithms such as greedy decoding, beam search, and top-k sampling with low k.

The effect of k This study detailed the typical characteristics of long-form text generated by neural language models in open-ended settings, under both high entropy (large k) and low entropy (small k) decoding algorithms. The negative characteristics of low k output (genericness, repetition, over-simplicity) are by now familiar to researchers. However, we also uncovered some less obvious characteristics of low-k generated text: compared to human-written text, it tends to copy more from the provided context (particularly GPT2-117); it contains more verbs and pronouns but fewer nouns and adjectives; its nouns are more concrete but its verbs are less concrete; and it uses a smaller range of syntactic patterns (a phenomenon that can't be entirely attributed to n-gram repetition).

Prompt: *In an alternative reality where sleep is non-existent among living beings, our protagonist (spontaneously or after an event) falls asleep in which he/she experiences for the first time in human history what a dream is.*

GPT2-117 ($k = 1000$**):** *I sat in my bed as my girlfriend sat behind me, buzzing into her e-reader, letting the day's stories write themselves on her's monitor. Like a blur, all the usual high-asyllabic drivel and senseless ramblings that normally attracted the attention of a horrid sleeping creature huddled about me like a faucet. She did not know how I placed this car with her. But I tried, first tried to ignore that I had hired the services of a dog to help and then quietly used it in a desperate bid to drive the car through the lawn. Each and every day, I watched her drool down an old dusty hardwood mattress her beady eyes trying desperately to think of this rotting dream.* [...]

Table 5.4: An example of syntactically natural but nonsensical text , generated with high k. Though the text is fluent and readable, it is surreal and bizarre.

As k increases to vocabulary size, we find that the model-generated text closely fits the human text on most of the metrics we measured. However, it is clear by inspection that the high-k model-generated text lacks many crucial aspects such as commonsense reasoning, world knowledge and multi-sentence coherence – an example of this superficially fluent but nonsensical text can be seen in Table 5.4. We believe that true progress in open-ended Natural Language Generation will come from attempting to address these high k problems – i.e., strategies to imbue the language model with better reasoning, knowledge and planning abilities – rather than continuing to seek ways to mitigate the diversity and repetition problems of the low k setting. Indeed, since this work was published, larger pretrained generative models such as GPT3 (Brown et al., 2020) and Meena (Adiwardana et al., 2020) appear to be able to use sampling-based decoding (with higher levels of k or p) to produce more unusual text without generating nonsense – see 'a note on scale' in Section 1.3.3.

Limitations of this study This study uses only the smallest version of GPT2, but it's likely that larger models may exhibit stronger statistical differences for the metrics we examine. Now that larger models are available, with even stronger generation quality, it would be very useful to see this study repeated. Such a study would illustrate the effect of larger model capacity, and more fully reveal the possible benefits of massive pretraining. We release our annotation code so that other researchers may repeat our study on more models and datasets.

While we investigated the full range of top-k decoding, we did not compare it to other decoding methods that adjust the quality-diversity tradeoff, such as top-p decoding and temperature adjustment (Section 2.6.2). Currently, all three methods are commonly used for open-ended neural text generation, without clear understanding of which provides the best tradeoff (Section 1.3.3). Exploring this question further would be very useful to the text generation community.

This study did not include human evaluation, which is currently the only reliable way to assess overall text quality, as well as quantify the deficiencies of high k output described above (coherence, reasoning, and world knowledge). As such, this study quantifies the *diversity* side more than the *quality* side of the quality-diversity tradeoff. Consequently, this study demonstrates the importance of developing better methods to computationally quantify notions such as text coherence, logicality and commonsense correctness – an effort that may ultimately hold the key to generating text with those desirable attributes.

Chapter 6

User Dissatisfaction in Chitchat Dialogue

6.1 Introduction

In Chapter 4, we found that carefully-controlled neural generative dialogue models are sometimes capable of conducting reasonable written conversations with crowdworkers, in a synthetic dialogue setting. Since that work, much larger models pretrained on more data have produced substantially better conversations when evaluated under similar conditions (Zhang et al., 2020b; Adiwardana et al., 2020; Roller et al., 2021) – though they still have problems with factual correctness (Mielke et al., 2020), using dialogue history (Sankar et al., 2019), and bias (Dinan et al., 2020).

By contrast, real-life settings such as the Alexa Prize (Gabriel et al., 2020), in which intrinsically-motivated users speak to open-domain chatbots in noisy environments, offer unique challenges. Unlike crowdworkers, users have their own expectations that may differ from those of the chatbot or its designers, and they may express dissatisfaction if those expectations are not met. It is not yet well-understood how neural generative models perform in these settings, nor the types and causes of dissatisfaction they encounter. By studying a neural generative model deployed in Chirpy Cardinal, an Alexa Prize chatbot, we seek to provide the first in-depth analysis of a neural generative model in large-scale real-life deployment, focusing on understanding the root causes of user dissatisfaction.

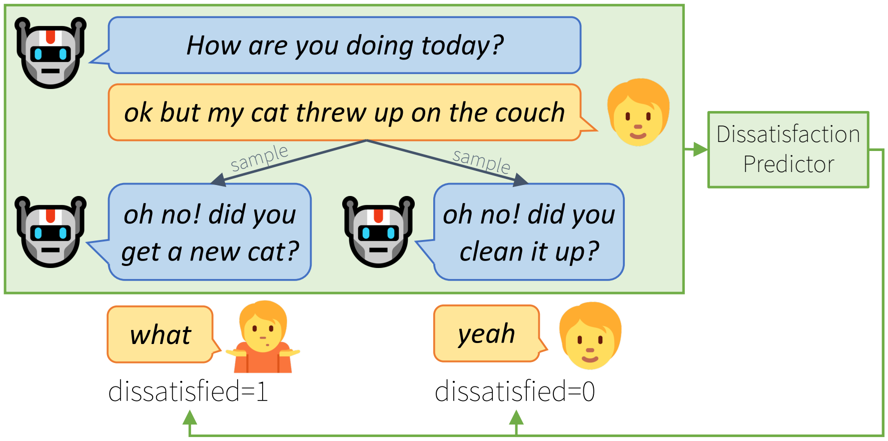


Figure 6.1: Users tend to express dissatisfaction (such as requests for clarification, left) after the neural generative chatbot makes errors (such as logical errors, left). Using past conversations, we train a model to predict dissatisfaction before it occurs. The model is used to reduce the likelihood of poor-quality bot utterances.

Real-life settings such as the Alexa Prize also offer unique opportunities. Dialogue systems can be difficult to build due to a lack of sufficient publicly-available data in the appropriate domain; meanwhile synthetic crowdsourced dialogue datasets can contain unnatural patterns or behaviors that are then replicated by a model trained on them. We use our chatbot's real-life conversations as a source of natural in-domain data. In particular, we train a model that can predict authentic user dissatisfaction before it occurs, thus helping us to avoid it.

Our Contributions. Through a detailed case-study of a neural generative model speaking with intrinsically-motivated users, we define taxonomies of neural generative errors and user dissatisfaction, and identify the relationships between them. We find that generative errors are common, though the noisy environment influences the rate and types of error. Our analysis suggests that improving commonsense reasoning and conditioning on history are high-priority areas for improvement. Though generative errors are correlated with user dissatisfaction, we find that the majority of errors do not immediately elicit user-expressed dissatisfaction, and some types of dissatisfaction (such as offensiveness and privacy objections) depend substantially on other factors, such as the user's own attitudes.

We then demonstrate a semi-supervised method to improve a neural generative dialogue system after deployment. We use an automatic classifier to silver-label dissatisfied user utterances in past conversations. Using these silver labels as training targets, we train another model to predict whether a given bot utterance will lead to user dissatisfaction (Figure 6.1). We show that this model is predictive of most dissatisfaction types, and when deployed as a ranking function, a human evaluation shows that it chooses higher-quality bot utterances.

6.2 Chirpy Cardinal

Chirpy Cardinal, aka CHIRPY (Paranjape et al., 2020; Chi et al., 2021)[1] is an open-domain socialbot developed for the Third and Fourth Alexa Prizes (Gabriel et al., 2020; Hu et al., 2021). The work in this chapter relates to the Third Alexa Prize version of CHIRPY (Paranjape et al., 2020).

During the Third Alexa Prize competition (December 2019 to June 2020), US Alexa customers could say *Alexa, let's chat* to connect to a random socialbot. Users would chat to the bot in English for as long as desired, then could provide a 1–5 rating. At the end of the competition, CHIRPY had an average rating of 3.6/5.0 and a median conversation duration of 2 minutes 16 seconds.

Like most Alexa Prize bots (Gabriel et al., 2020), CHIRPY is modular in design, combining a mix of rule-based, retrieval-based, knowledge-based and neural generative components specializing in different topics. However, this chapter focuses solely on the Neural Chat module, which uses neural generation. An open-source version of CHIRPY is available, including the code and pretrained model for the Neural Chat module.[2]

6.2.1 Neural Chat Module

Discussion areas and starter questions. Empathy is a fundamental part of human communication, and can improve user experience of dialogue agents (Ma et al., 2020). The Neural Chat module aims to offer an empathetic experience by showing an interest in the

[1] https://stanfordnlp.github.io/chirpycardinal

[2] https://github.com/stanfordnlp/chirpycardinal

user's feelings and experiences. It has seven discussion areas, all relating to personal experiences and emotions: Current and Recent Activities, Future Activities, General Activities, Emotions, Family Members, Living Situation, and Food. A Neural Chat discussion begins by asking the user a handwritten starter question from one of the discussion areas; these are designed to be easy-to-answer and applicable to most users, and are listed in Section 6.2.2.

Neural model and training. For subsequent turns of the discussion, responses are provided by a GPT2-medium (Radford et al., 2019) model with 345 million parameters – this is larger than the 117 million GPT2-small version used in Chapter 5. Though even larger GPT2 models were available at the time of building the Neural Chat module, their latency and cost was prohibitively high for inclusion in CHIRPY.

Our GPT2-medium model is finetuned on the EmpatheticDialogues dataset (Rashkin et al., 2019). The dataset consists of conversations between a *speaker*, who describes an emotional personal experience, and a *listener*, who responds empathetically to the speaker's story. Our model is trained in the listener role.

Generating and selecting neural responses. On each turn, we provide the current Neural Chat discussion history as context to the GPT2 model, and generate 20 possible responses using top-p sampling with $p = 0.9$ and temperature 0.7. Repetitive responses (containing previously-used trigrams) are removed. Except when transitioning out of the Neural Chat discussion (see below), we always choose a neural response containing a question.[3] Of the responses satisfying these criteria, we choose the longest response, as it tends to be the most substantive and interesting.

Ending discussions. A Neural Chat discussion can end in several ways. The user may initiate a topic better handled by another CHIRPY module (*what do you know about baseball*), or express dissatisfaction (see Section 6.3), in which case another CHIRPY module will take over. Otherwise, if under a third of the sampled Neural Chat responses contain questions, we interpret this as a heuristic indication that the model is not confident in asking a question

[3]Many Alexa Prize bots end most utterances with a question (Gabriel et al., 2020). We found that users were unsure what to say if the bot did not offer a clear direction. However, constant questions can fatigue users (Paranjape et al., 2020), and reducing question-asking can increase user initiative (Hardy et al., 2021).

on this turn. In this case, we choose a non-question, and transition to a different CHIRPY module. Paranjape et al. (2020) provides full details of the Neural Chat module and how it fits into CHIRPY.

Under this strategy, each Neural Chat discussion contains a mean of 2.75 bot utterances. While this is shorter than ideal, we found that if we extended the Neural Chat conversations, after a few turns the bot would often give a poor-quality response that would derail the conversation. The brevity of the Neural Chat discussions limits its conversational depth, and thus its ability to provide the desired empathetic user experience. The rest of this chapter focuses on understanding what kinds of poor-quality neural responses derail the discussions, and how we can learn to avoid them.

6.2.2 Starter Question Examples

This section provides examples of starter questions used in the Neural Chat module's discussion areas (Section 6.2.1). A full list can be found in the open-source release of CHIRPY.[4]

Current and Recent Activities Questions typically reference the day of the week, then ask a question depending on the user's time of day:

- *It's a beautiful Saturday here in the cloud. What are your plans for the rest of today?* (morning)
- *I hope you're having a wonderful Monday. What did you do today?* (evening)

Future Activities The question depends on the day of the week and the user's time of day:

- *It's the weekend soon! Do you have any plans for the weekend?* (Friday)
- *Before I go to bed I like to think about something I'm looking forward to tomorrow. What about you, are you doing anything nice tomorrow?* (9pm–2am)

General Activities

[4] https://github.com/stanfordnlp/chirpycardinal

- *Recently, I've been trying meditation to help me relax during this stressful time. What do you like to do to relax?*
- *I was reading earlier today that staying busy helps people stay calm and healthy during stressful times. What do you like to do to keep busy?*

Emotions The starter question *I hope you don't mind me asking, how are you feeling?* is preceded by several possible preambles, that might involve the bot sharing its own (negative or positive) feelings, and/or a personal anecdote.

- *I wanted to check in with you. I hope [..] feeling?*
- *I wanted to say that I'm feeling pretty positive today! I hope [..] feeling?*
- *I wanted to say that I've been feeling kind of down recently. I've been missing my friends a lot and finding it hard to focus. I hope [..] feeling?*

Family Members This area is triggered if the user mentions one of several predefined phrases referring to family members (e.g., parents, grandparents, siblings, cousins, children), friends, or pets. Questions depend on the type of family member:

- *You mentioned your parents. I'd love to hear more about them, if you'd like to share. How did they meet?*
- *You mentioned your dog. I'd love to hear more about them, if you'd like to share. What kind of dog do you have?*

Living Situation This area is targeted at living experiences during the COVID-19 pandemic:

- *It seems that a lot of people are finding the quarantine lonely, and other people can't get enough space away from their families or roommates. What's it been like for you?*

Food Depending on the user's time of day, questions typically ask about a meal that is likely to be upcoming or recently eaten:

- *It's breakfast time, my favorite time of day! What are you having for breakfast today?*
- *I hope you're having a wonderful evening. What did you have for dinner today?*

Dissatisfaction Type	Definition	Examples	Freq.
Clarification	Indicates the bot's meaning isn't clear	*what do you mean, i don't understand what you're talking about*	2.28%
Misheard	Indicates the bot has misheard, misunderstood or ignored the user	*that's not what i said, you're not listening to me*	0.24%
Repetition	Indicates the bot has repeated itself	*you already said that, we talked about this already*	0.03%
Criticism	Expresses a critical opinion of the bot	*you're so rude, you're bad at this, you're not smart*	0.56%
Privacy	Indicates the bot has overstepped a privacy boundary	*none of your business, why are you asking me that, you're being creepy*	0.11%
Offensive	Contains obscene/offensive words or topics	*will you talk dirty, what size are your boobs, stick it up your ass*	1.54%
Negative Navigation	Expresses desire to end current topic	*change the subject, i don't want to talk about this*	0.59%
Stop	Expresses desire to end conversation	*i have to go bye bye, end the conversation please*	3.68%
Any	Expresses one or more of the above	Any of the above examples	11.56%

Table 6.1: User dissatisfaction types. Frequency of type D is estimated by the proportion of NeuralChatTurns examples (c, b, u) where the k-NN classifier for D assigns u a score of 0.5 or more: $P_{\text{kNN}}(D|u) \geq 0.5$.

6.3 Detecting User Dissatisfaction

We consider a user utterance to express *dissatisfaction* if it meets any of the definitions in Table 6.1. An utterance can express multiple types of dissatisfaction; e.g., *what do you mean stop* is both Clarification and Stop. Though some types, such as Stop, might not necessarily represent dissatisfaction (as every user must eventually end the conversation) these dissatisfaction types are strong indicators that the bot has recently given a poor-quality response.

Regex classifiers In CHIRPY, we manually designed regex classifiers to identify each of the dissatisfaction types in Table 6.1.[5] If a user utterance triggers one of these classifiers, CHIRPY takes the appropriate action (e.g., ending the conversation, switching topic, apologizing). The classifiers are designed to capture the most commonly-expressed forms of each

[5]The regexes are in the CHIRPY open-source code: https://github.com/stanfordnlp/chirpycardinal

dissatisfaction type; they are high precision but lower recall (Paranjape et al., 2020).

Human-labelled set To help us develop higher recall dissatisfaction classifiers, one expert annotator[6] gathered a set of 3240 user utterances. For each utterance u and dissatisfaction type D, they provided a label $\text{HumLabel}_D(u) \in \{0, 1\}$. The utterances are drawn from several sources, including most common utterances, utterances drawn from 1-rated conversations, and utterances which scored highly for the *clarifying*, *closing* and *complaint* dialogue acts in CHIRPY's Dialogue Act classifier (Paranjape et al., 2020).[7]

Nearest Neighbors classifiers To represent a user utterance u, we take a DialoGPT-large model (Zhang et al., 2020b) that was finetuned on CHIRPY conversations,[8] input u, and average the top-layer hidden states across the sequence. Using this embedding for each utterance, we build a FAISS (Johnson et al., 2017) index of the human-labelled set. To compute a new utterance u's score for dissatisfaction type D (including Any), we find its k Nearest Neighbors $u'_1, ..., u'_k$ in the human-labelled set (w.r.t. cosine distance), then compute $P_{\text{kNN}}(D|u) \in [0, 1]$ as follows:

$$P_{\text{kNN}}(D|u) = \begin{cases} \text{HumLabel}_D(u) & \text{if } u \text{ human-labelled} \\ 1 & \text{if } u \text{ matches } D\text{-regex} \\ \frac{1}{k}\sum_{j=1}^{k} \text{HumLabel}_D(u'_j) & \text{otherwise.} \end{cases} \tag{6.1}$$

That is, we first check if u has a human label or is a positive match for D's regex; if not we compute the proportion of u's neighbors that are labelled D.

For each D, we evaluate the k-NN classifier on the human-labelled set for $k = 1, \ldots, 30$

[6]Due to privacy constraints, Alexa Prize user conversations can only be viewed by official team members. Thus all annotators in this chapter are team members, not crowdworkers.

[7]These sources were chosen to obtain a greater proportion of dissatisfied examples; this increases the sensitivity of the human-labelled set without needing to label a very large set.

[8]The CHIRPY conversations comprise 1.2GB of text data, collected over the competition. We separate utterances with the <|endoftext|> token (as DialoGPT was trained), and divide the data into chunks of 256 tokens. Using Huggingface Transformers (Wolf et al., 2020), we trained on a Titan RTX for 1 epoch (more led to overfitting), with batch size 4, 2 gradient accumulation steps, Adam optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, ϵ =1e-8, and initial learning rate 5e-5. The DialoGPT-large model reached a perplexity of 2.17 on the CHIRPY validation set (2.30 for DialoGPT-medium, 2.58 for DialoGPT-small).

Dissatisfaction Type	Optimal k	AUPRC ↑
Clarification	10	0.616
Misheard	26	0.474
Privacy	8	0.504
Repetition	4	0.476
Criticism	28	0.647
Negative Navigation	4	0.492
Offensive	5	0.705
Stop	4	0.828
Any	7	0.787

Table 6.2: Performance (AUPRC) of k-NN dissatisfaction classifiers on the human-labelled set (Section 6.3).

via leave-one-out cross-validation. Table 6.2 shows the optimal k and area under the precision-recall curve (AUPRC) for each D.

6.4 NeuralChatTurns Dataset

Over the period that CHIRPY was online, we collect examples of the form (c, b, u) where b is a purely neural-generated bot utterance, c is the Neural Chat context that preceded b, and u is the user response to b. The NeuralChatTurns dataset has 393,841 examples in total, which we split into 315,072 train, 39,384 validation, and 39,385 test. Due to user privacy constraints, we are not permitted to publicly release the NeuralChatTurns dataset.

6.5 Annotation Details

To understand dissatisfaction, we annotate errors in the generative model's conversations. By inspecting the neural-generated output, we develop a taxonomy of bot errors; these are defined in Table 6.3 with examples in Section 6.5.1. In addition to bot errors, we consider two other potential causes of dissatisfaction: first, whether the user is already dissatisfied in the Neural Chat context c; second, whether the user's utterance is clear. Unclear user utterances – caused by ASR errors, misspeaking, ambiguity, or background noise – present challenges in CHIRPY (Paranjape et al., 2020) and across the Alexa Prize (Gabriel et al., 2020).

From the NeuralChatTurns validation set, we randomly sample a control set of 100 (c, b, u) examples, and annotate u's dissatisfaction types. As dissatisfaction is relatively rare (Table 6.1), for each dissatisfaction type D we additionally gather 100 (c, b, u) examples where u is of type D.[9] For these 900 (c, b, u) examples, one expert annotator viewed each (c, b) example (without seeing u), and annotated it for the problems in Table 6.3. As the bot error types are somewhat subjective, we collected some additional second annotations to measure inter-annotator agreement (see Section 6.5.2). Annotators were provided the definitions in Table 6.3 and the examples in Section 6.5.1.

6.5.1 Bot Error Examples

User unclear

- User hesitated/cutoff: *What did you have for dinner today?* $\rightarrow$ *well*
- ASR errors / misspeaking has corrupted user utterance: *What are your plans for today?* $\rightarrow$ *maybe go to cost coat*
- Answering yes/no to a non-yes/no question: *What do you like to do to relax?* $\rightarrow$ *yes*

Bot repetitive

- Asking same essential question, even if reworded: *What are your plans today?* $\rightarrow$ *i don't know* $\rightarrow$ *I'm sorry to hear that. What will you do?*
- Repeating question, even if we didn't get a clear answer the first time: *What are your plans today?* $\rightarrow$ *it's not monday* $\rightarrow$ *What will you do today?* This is **repetitive but not redundant**.

Bot redundant question

- Repeating a question the user already answered: *What are your plans today?* $\rightarrow$ *i don't know* $\rightarrow$ *I'm sorry to hear that. What will you do?*

[9] To obtain these, we sample (c, b, u) where $P_{\mathrm{kNN}}(D|u) > 0$ without replacement, and manually verify until we have 100.

Problem	Definition	% in ctrl set	% when no user prob.
User already dissatisfied	The user has already expressed dissatisfaction in c.	12.0%	0.0%
User unclear	The main gist of the user's latest utterance in c is unclear or obscured.	22.0%	0.0%
Bot repetitive	The primary content of b was already said/asked by the bot earlier in c.	6.0%	4.3%
Bot redundant question	b is asking for information that the user has already provided earlier in c.	12.0%	15.9%
Bot unclear	It's hard to find an interpretation of b that makes sense.	12.0%	7.2%
Bot hallucination	b refers to something that hasn't been mentioned, acts like the user said something they didn't, confuses self with user, or seems to be responding to own utterance.	17.0%	10.1%
Bot ignore	b ignores or fails to acknowledge the user's latest utterance, doesn't answer a question, doesn't adequately respond to a request, or switches to an unrelated topic.	20.0%	14.5%
Bot logical error	b is generally on-topic, but makes an assumption or association that's incorrect, unfounded or strange.	15.0%	17.4%
Bot insulting	b says or implies something insulting about the user, or about others in a way that might offend the user.	1.0%	1.4%
Any bot error	True iff any of the above *bot* errors are true.	53.0%	46.4%

Table 6.3: Definitions of problems that may be present in a NeuralChatTurns example ($c =$ context, $b =$ bot utterance); prevalence in the control set ($n = 100$); prevalence in control set examples with no user problems ($n = 69$).

- Asking for info the user has provided, even if the question is new: *What did you do today?* $\rightarrow$ *i played with my sister* $\rightarrow$ *Oh cool! Do you have a sister?* This is **redundant but not repetitive**.

Bot unclear

- Hard to understand or answer bot's question: *How are you feeling?* $\rightarrow$ *what feeling feeling i'm* $\rightarrow$ *What is it?*

- Most bot hallucination examples
- Poor fluency or phrasing obscures bot's meaning: *What do you like to do to keep busy? → reading → I like to read, watch movies, do any sports?*

Bot hallucination

- Referring to something not mentioned: *What's quarantine been like for you? → ok actually → That's good. How long have you been together?*
- Acting like the user said something they didn't: *What did you have for lunch? → well i → Oh cool, what kind of sandwich?*
- Confusing self with user: *What did you do today? → uh → I did what?*
- Responding to own utterance: *I'm feeling kinda down. How are you? → ok → I'm sorry to hear that, what's bothering you?*

Bot ignoring

- Ignore / fail to acknowledge user's utterance: *What are some recipes I should cook? → maybe pasta → I love cooking, do you?*
- Doesn't answer a question: *What are your plans for the weekend? → do you have a boyfriend → a boyfriend? how long have you been together?*
- Doesn't adequately respond to a request: *What are your plans for the weekend? → alexa play music → I love music. Do you like live concerts?*
- Switching to unrelated topic: *What are your hobbies? → painting → How about fishing?*

Bot logical error

- Question makes sense, but is strange: *Tell me about your mom → she's cool → is she a nurse?*
- Reacting the wrong way (e.g., wrong sentiment): *What are you doing tomorrow? → visiting grandma → I'm sorry to hear that.*
- Bot is inconsistent with itself: *I've been feeling down. How are you feeling? → great how about you → I'm doing great too.*
- Mixing up different people in the scenario: *Tell me about your sister → she's at college → Oh cool, what's your major?*
- Wrong word sense: *What did you today? → i played tag → What kind of name tag?*

Problem	**Freq.**	**Scott's pi (unclear/all/clear)**
User unclear	35.7%	- / 0.70 / -
Bot repetitive	20.1%	0.50 / 0.62 / 0.72
Bot redundant q.	15.6%	0.19 / 0.50 / 0.58
Bot unclear	16.4%	0.45 / 0.52 / 0.56
Bot halluc.	31.6%	0.35 / 0.45 / 0.43
Bot ignore	25.8%	-0.13 / 0.34 / 0.59
Bot logical err.	23.0%	0.02 / 0.17 / 0.27
Bot insulting	5.7%	-0.04 / 0.24 / 0.35
Any bot err.	75.0%	0.08 / 0.45 / 0.68

Table 6.4: Frequency and inter-annotator agreement for each label.

Bot insulting

- Saying/implying something insulting about the user: *How are you feeling?* $\rightarrow$ *sad* $\rightarrow$ *Oh, I suppose you have no friends?*
- Rudely saying/implying the user should do something: *What did you do today?* $\rightarrow$ *nothing* $\rightarrow$ *I see. Why don't you get a job?*
- Expressing a negative opinion of someone (beyond just sympathizing/acknowledging user's opinion): *Tell me about your brother.* $\rightarrow$ *he's always bugging me* $\rightarrow$ *He sounds so annoying.*

6.5.2 Inter-annotator Agreement

For 122 randomly-selected examples annotated by the first annotator, we collected annotations from a second annotator. Table 6.4 shows the frequency of each label (among the pooled 244 judgments), and Scott's pi agreement (Scott, 1955), divided into unclear examples (where at least one annotator judged the user utterance unclear), all examples, and clear examples (where both annotators judged the user utterance clear). In all cases, agreement is higher when the user utterance is clear. We found bot errors harder to diagnose when the user's utterance is unclear – e.g., if the user's utterance is completely nonsensical, what does it mean for the bot to adequately acknowledge it?

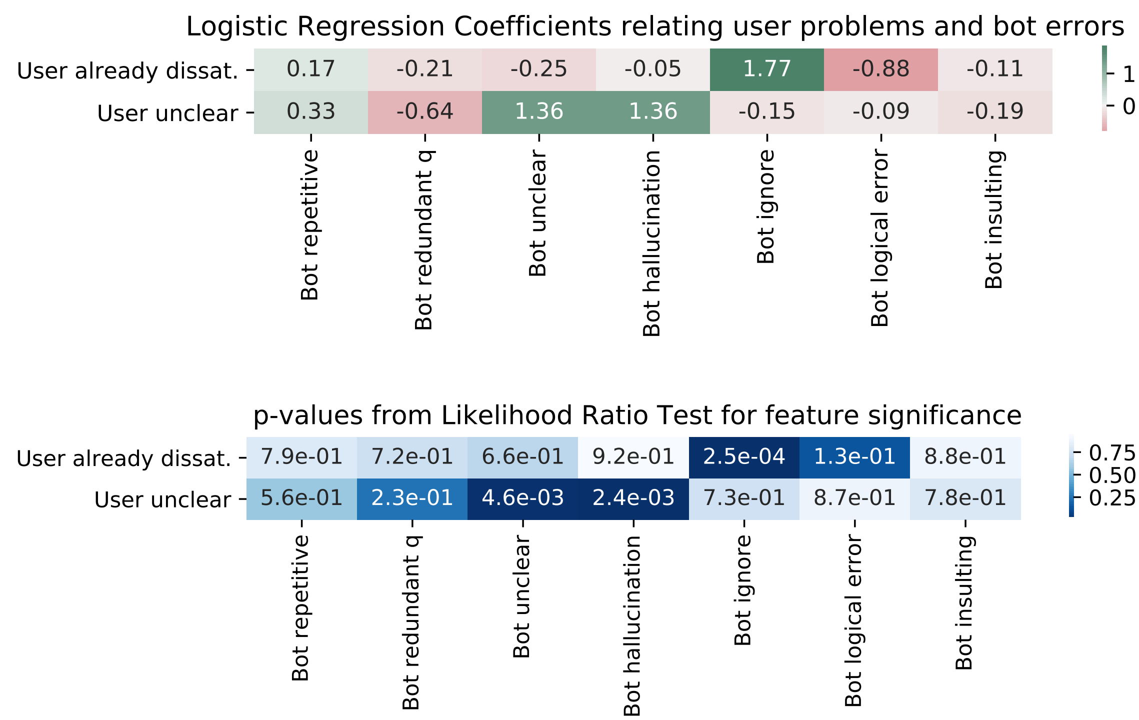


Figure 6.2: For each bot error E, we use the control set (Section 6.5) to fit a Logistic Regression model to predict E using the two rows above as features. For each feature we perform a Likelihood Ratio Test to determine if including that feature results in a statistically-significant improvement to the model's fit.

6.6 What Causes User Dissatisfaction?

6.6.1 Effect of Unclearness and Prior Dissatisfaction on Bot Errors

Table 6.3 shows that the user's utterance is unclear in 22% of control set examples. In these contexts, it's impossible for the bot to reliably produce a good response. Indeed, Figure 6.2 shows that unclear user utterances are significantly ($p < 0.05$) predictive of bot hallucinations and unclear bot utterances. In practice, we observe that when the user's utterance is unclear, the generative model tends to hallucinate (in many cases, responding as if the user had said something more expected), or respond unclearly (often, this is a vague question such as *What is it?*) – examples of both are in Section 6.5.1.

Table 6.3 also shows that, in 12% of examples, the user has already expressed dissatisfaction in the Neural Chat context c. Ordinarily, the regex-based dissatisfaction classifiers should detect dissatisfaction and interrupt the Neural Chat conversation to handle it (see Section 6.3) – thus these examples represent false negatives of the regex classifiers. As the generative model is generally unable to adequately respond to dissatisfaction (e.g., requesting to stop the conversation), most of these examples are also impossible for the generative model to handle. Accordingly, we find a significant positive relationship between prior user dissatisfaction and bot ignoring (Figure 6.2).

Nevertheless, after removing these user problems, bot errors are still common: for the 69 control set examples where the user is clear and not already dissatisfied, 46.4% of bot utterances contain at least one type of error (down from 53% in the whole set; see Table 6.3). Among these examples, the more basic errors (repetitive, unclear, hallucination, ignoring) become less common, and the errors relating to reasoning or social abilities (redundant, logical, insulting) are more common.

6.6.2 Effect of Bot Errors on User Dissatisfaction

Despite the high rate of bot errors in the control set (53 in 100), only a minority of users express dissatisfaction immediately after an error (8 in 53; 15%). In fact, we observe that some users respond to errors by helpfully teaching CHIRPY about the world – e.g., *you pick things up and put them away* to explain the concept 'cleaning your room'.

Figure 6.3 shows the contribution (as a logistic regression coefficient) of each problem in Table 6.3 to each dissatisfaction type. We find that each bot error (except logical error[10]) is significantly ($p < 0.05$) predictive of at least one dissatisfaction type. We find that bot repetition is the least-tolerated error, being significantly predictive of six dissatisfaction types. Other than bot repetition, the likelihood of ending the conversation (NegNav/Stop) is significantly raised by unclear bot utterances – perhaps because it becomes impossible to continue the conversation – and by bot insults. Other positive relationships include unclear user with Misheard, repetitive and redundant bot with Repetition, unclear bot with

[10]This exception may be because by definition (Table 6.3), logical errors tend to occur in the absence of more basic errors (such as repetition, unclear, ignoring, and hallucination) so are less likely to completely derail the conversation.

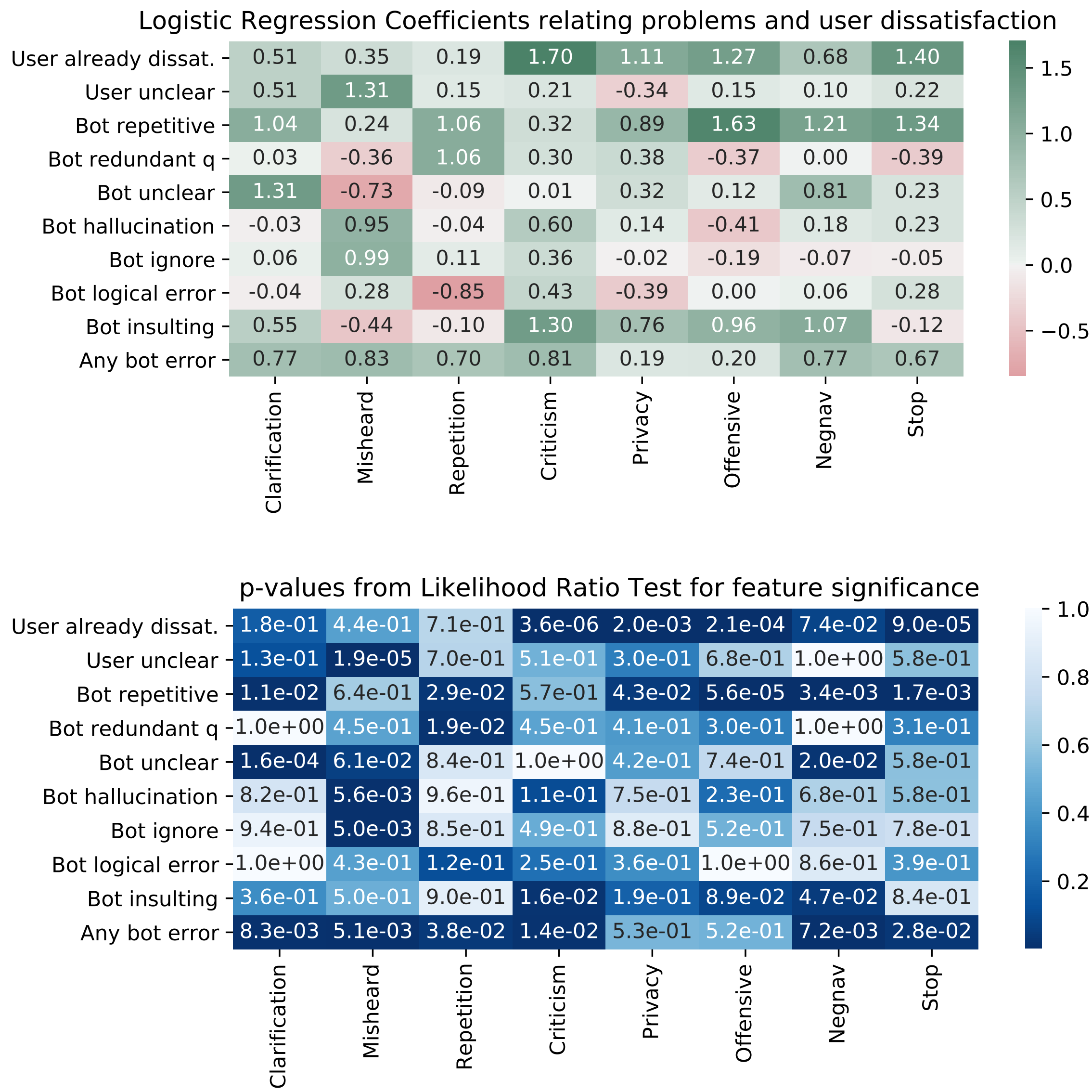


Figure 6.3: For each dissatisfaction type D, we take the 100 control examples plus the 100 D examples (Section 6.5), and fit a Logistic Regression model to predict D using the first 9 rows above as features. To obtain the values in the *Any bot error* row, we use just the first two and last row as features. For each feature, we use a Likelihood Ratio Test to determine if including that feature results in a statistically-significant improvement to the model's fit.

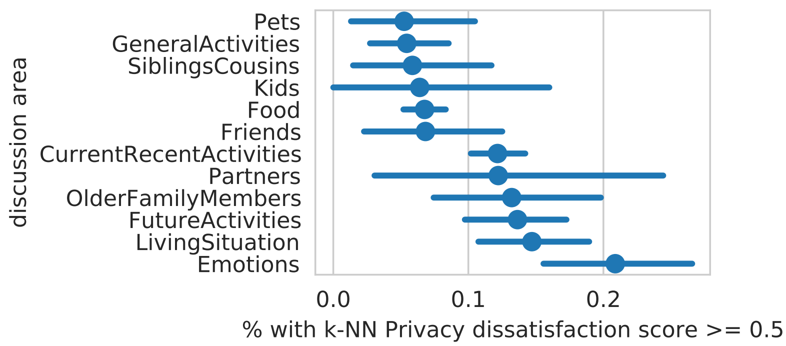


Figure 6.4: Privacy dissatisfaction rate (with 95% CIs) for each Neural Chat discussion area (see Section 6.2.2).

Clarification, bot hallucination and ignoring with Misheard, and bot insulting with Criticism.

Six of the eight dissatisfaction types have a significant positive correlation with Any bot error. Privacy is least-correlated with bot errors; this makes sense, as privacy boundaries are extremely subjective (Section 6.6.4). Offensive is next least-correlated, reflecting that offensive users can be motivated by factors other than poor bot performance – e.g., a curiosity to test the bot (De Angeli et al., 2005; De Angeli and Brahnam, 2008). Repetition has the third weakest correlation; indeed, we find that 28% of Repetition complaints occur in the absence of an annotated bot error. These users may be complaining about the bot repeating something from outside the Neural Chat context c, or something said by a different Alexa Prize bot.

6.6.3 Unaddressed Dissatisfaction Escalates

Figure 6.3 shows that prior user dissatisfaction is significantly ($p < 0.05$) predictive of several types of subsequent dissatisfaction. We recompute this analysis for two cases: with and without a bot error. Among bot error examples, we find prior dissatisfaction is significantly correlated with Criticism, Stop, Privacy, and Offensive – indicating that already-dissatisfied users are more likely to respond to bot errors with complaining, quitting, or offensiveness. Among examples without a bot error, prior dissatisfaction is significantly correlated with Offensive – indicating that already-dissatisfied users are more likely to be offensive, even in response to a good-quality bot utterance.

6.6.4 Privacy Boundaries Vary

Self-disclosure – the act of revealing information about oneself – is an important part of developing and maintaining relationships (Dindia, 2002). It is also reciprocal; meaning that people are more inclined to disclose if their conversational partner has already done so (Dindia, 2002). Indeed, we find that if CHIRPY proactively self-discloses – sharing its feelings and experiences before asking the user's – users give longer responses on average (Paranjape et al., 2020).

When talking to chatbots, users can have varying and complex attitudes to self-disclosure. Croes and Antheunis (2020) report that chatbots are perceived as more anonymous and non-judgmental than humans; this can increase user self-disclosure. However, some users perceive chatbots as lacking trust and social presence, inhibiting user self-disclosure. In CHIRPY, we observe both phenomena – some users share their thoughts and feelings candidly, while others react with suspicion (e.g., *are you spying on me*) to questions typically regarded as appropriate between strangers in US society (*What are you up to today?*).

Figure 6.4 shows that emotional topics (including Living Situation, see Section 6.2.2) are most likely to be rejected on privacy grounds. Users are more comfortable discussing general activities (e.g., *What are your hobbies?*) than specific activities in the present or future (*What are your plans for the weekend?*). For the Family Members discussion area, users are more comfortable discussing pets, siblings, kids and friends, and less comfortable discussing partners and older generations.

6.7 Learning to Predict User Dissatisfaction

In this section we build a system to predict, and thus reduce the likelihood of, dissatisfaction.

6.7.1 Predictor Training Details

We take a DialoGPT-large model (Zhang et al., 2020b) that was finetuned on CHIRPY conversations, and finetune it on NeuralChatTurns training examples (c, b, u) as follows. The input to the model is a context and bot utterance (c, b), with the utterances separated by the `<|endoftext|>` token. We wish to predict $P_{\text{pred}}(\text{Any}|c, b)$, the probability that

Dissatisfaction	**Predictor correlation** $\rho \uparrow$	p**-value**
Clarification	0.274	8.7e-05
Misheard	0.295	2.2e-05
Repetition	-0.038	6.5e-01
Criticism	0.429	2.2e-10
Privacy	0.326	3.5e-06
Offensive	0.394	7.7e-09
Neg. nav.	0.204	3.8e-03
Stop	0.209	3.0e-03

Table 6.5: Spearman correlation between predictor output and each human-annotated dissatisfaction type D (computed on 100 control and 100 D examples).

the next user utterance u will express Any dissatisfaction. To compute this, we take $H_{L,t} \in \mathbb{R}^{1280}$, the hidden state of the top-layer L for the last timestep t of the input, and apply a linear layer ($W \in \mathbb{R}^{1280}$) and sigmoid activation:

$$P_{\text{pred}}(\text{Any}|c, b) = \sigma(W^T H_{L,t}) \in [0, 1] \tag{6.2}$$

We train the predictor with Mean Squared Error to match the probability that u expresses Any dissatisfaction, as given by the k-NN classifier:

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} \left(P_{\text{pred}}(\text{Any}|c_i, b_i) - P_{\text{kNN}}(\text{Any}|u_i)\right)^2 \tag{6.3}$$

$P_{\text{kNN}}(\text{Any}|u_i)$ is as defined in Section 6.3, using the optimal k for Any (Table 6.2). We finetuned the DialoGPT-large-CHIRPY model for 1 epoch (more led to overfitting) with the same hardware and hyperparameters as the DialoGPT-large in Section 6.3 (except learning rate 2e-05). The DialoGPT-large-CHIRPY model reached a MSE of 0.0727 on the NeuralChatTurns validation set (0.0728 for without CHIRPY pretraining).

6.7.2 How Accurately Does the Predictor Predict Dissatisfaction?

On the NeuralChatTurns validation set, the predictor's output and the P_{kNN} targets have a Spearman correlation $\rho = 0.30$.[11] This indicates a statistically significant but noisy

[11] $p <$1e-5, Fisher transformation test (null hypothesis ρ=0)

correlation between the predictor's output and the automatically-provided targets. With respect to the *human*-provided labels for Any dissatisfaction (Section 6.5), the predictor has a a similar correlation of $\rho = 0.28$ ($p = 0.0043$). This indicates that the difference between the true dissatisfaction labels and the P_{kNN} training estimates is not a primary limitation of the predictor's accuracy.

Table 6.5 shows that the predictor has significant ($p < 0.05$) positive correlation with each dissatisfaction type except Repetition. This may be because Repetition is the rarest type in the training set (Table 6.1), or because some Repetition complaints are not predictable from the Neural Chat context (Section 6.6.2).

6.7.3 What Information Does the Predictor Use?

First, we perform an ablation analysis. Compared to the full model's correlation of $\rho = 0.30$ with the P_{kNN} targets, the predictor achieves $\rho = 0.25$ if trained only on the context c, and $\rho = 0.23$ if trained only on the bot utterance b (all $p <$1e-5).

Separately, on the human-annotated control set we find that the full predictor model has a positive correlation $\rho = 0.26$ ($p = 0.0087$) with prior user dissatisfaction, a weaker correlation $\rho = 0.21$ ($p = 0.035$) with unclear user utterance, and no significant correlation with the presence of any bot problem: $\rho = 0.022$ ($p = 0.83$).

Together this evidence indicates that the predictor learns to condition more strongly on c (in particular prior user dissatisfaction) and less on b (in particular bot errors). Though concerning, this is unsurprising, as user dissatisfaction (which we can detect automatically) is simpler to detect than bot errors (which require human annotation).

However, as evidenced by the b-only ablation result, the predictor does find some useful signal in b. In particular, we find that the full model conditions strongly on the bot's question. Figure 6.5 (top) shows that in NeuralChatTurns data, *What happened?*, *What are you doing?* lead to more dissatisfaction,[12] whereas positive questions such as *Did you have fun?*, *Did you enjoy it?* tend to lead to less. Figure 6.5 (bottom) shows that the predictor learns these patterns quite closely.

[12] These questions are often used repetitively, if the user's answer to the first asking is unclear/negative (see Section 6.5.1).

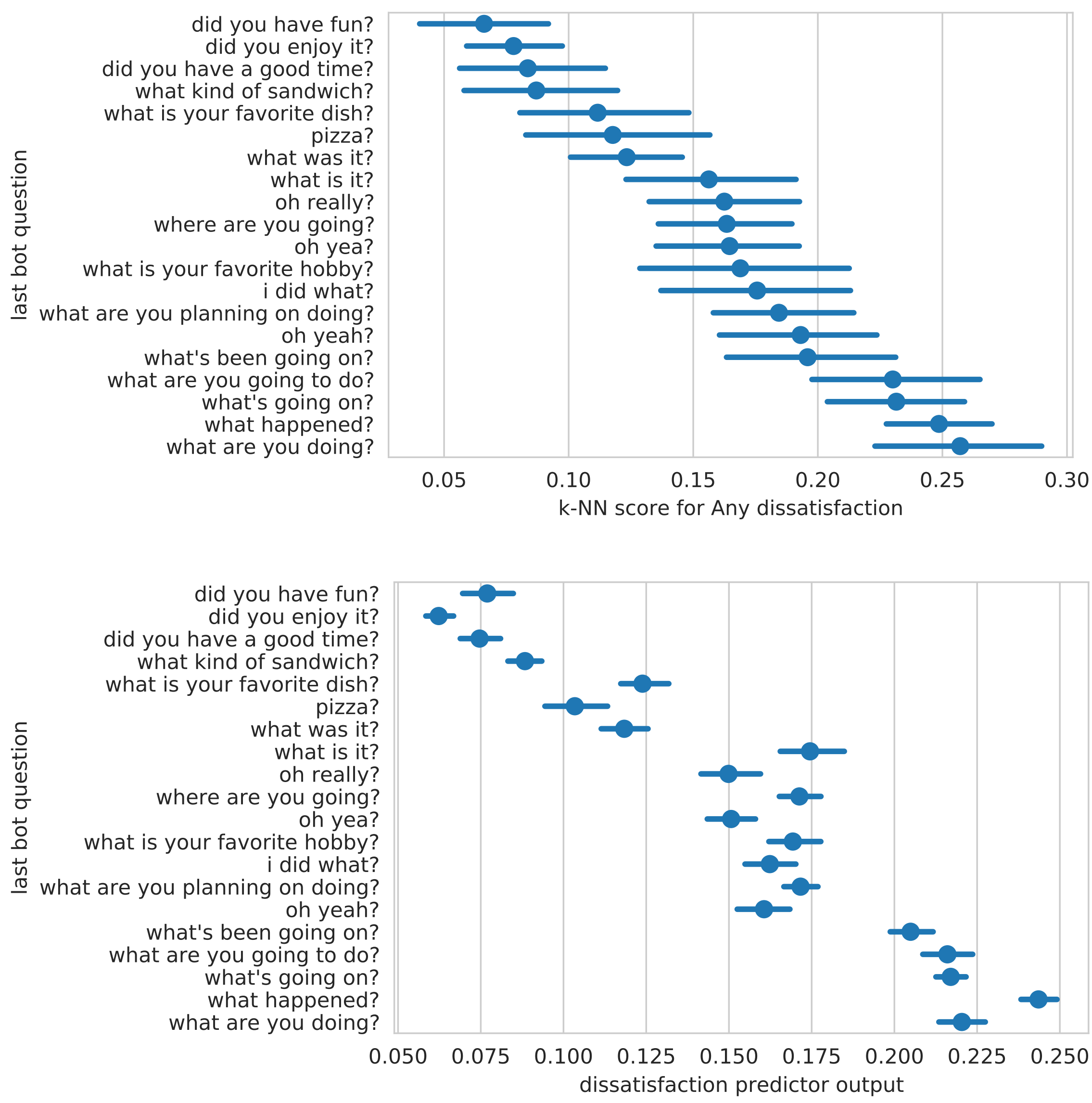


Figure 6.5: For each of the 20 most common bot questions, mean scores and 95% CIs for Any dissatisfaction given by the k-NN classifier (top) and the predictor (bottom).

6.8 Ranking Bot Responses to Minimize Dissatisfaction

Section 6.7 showed that the predictor learned a positive, though noisy, correlation with user dissatisfaction. In this section we use the predictor to select better-quality bot utterances that are less likely to lead to user dissatisfaction.

6.8.1 Human Evaluation Details

Given that the generative model is generally incapable of responding well when the user is unclear or already dissatisfied, we focus on improving its performance on the remaining cases (which we call *achievable*). We sample 400 examples from the NeuralChatTurns validation set, then manually filter to obtain 270 achievable examples. For these, we take the context c and generate 20 possible bot responses $b_1, \ldots, b_{20}$, using the generative model and decoding procedure in Section 6.2.1. Let b_{pred} be the response with best (i.e., lowest) predictor score: $b_{\text{pred}} = \text{argmin}_{b_j \in b_1,\ldots,b_{20}} P_{\text{pred}}(\text{Any}|c, b_j)$. We randomly sample an alternative b_{rand} uniformly from the other 19 responses. One expert evaluator viewed each c, then chose which of b_{pred} or b_{rand} (presented blind) is a higher-quality response. If only one of the two has an error (defined in Table 6.3), the non-error response is preferred. If neither or both have an error, the response that better responds to the user's utterance and continues the conversation is deemed higher-quality.

6.8.2 Results

We find that b_{pred} is preferred in 46.3% of cases, b_{rand} in 35.6%, and no preference in 18.1%. A binomial test (null hypothesis: b_{pred} and b_{rand} equally likely to be preferred) returns a p-value of 0.03. This raises the question: if the predictor's outputs have no significant correlation with bot errors in the NeuralChatTurns distribution (Section 6.7.3), how does the predictor select better-quality bot utterances on average? Section 6.7.3 showed that the predictor *does* condition on b, in particular the bot question, but it conditions on c more strongly. It's possible that when $c_i = c_j$ (as in this evaluation), the predictor is able to distinguish quality differences between (c_i, b_i) and (c_j, b_j); however, on the NeuralChatTurns dataset where the c_i and c_j are distinct, the effect of c_i and c_j dominates the predictor's ranking.

6.9 Related Work

Previous work has used a variety of user signals to improve dialogue agents. When learning from a variable-quality human-human dataset such as Reddit, Gao et al. (2020) showed that engagement measures like upvotes and replies are more effective than perplexity to train a ranking model. For one-on-one empathetic conversations like ours, Shin et al. (2019) trained a neural generative model with reinforcement learning to improve next-turn user sentiment (as simulated by a user response model, rather than human responses). Though we considered taking a sentiment-based approach in CHIRPY, we found that user sentiment doesn't always align with good user experience: first, expressing negative emotions is sometimes unavoidable, and second, sentiment classifiers tend not to distinguish between sentiment about the conversation and sentiment about other issues. We find next-turn user dissatisfaction to be a comparatively more precise, well-aligned learning signal.

Dialogue systems that learn from their *own* interactions with humans are relatively rare. Hancock et al. (2019) also use user satisfaction to identify high-quality bot utterances; these become additional training examples for the neural generative model. However, Hancock et al. use paid crowdworkers; research involving intrinsically-motivated, unpaid users is rarer still. In symmetric settings such as the role-playing game LIGHT (Shuster et al., 2020), the user utterances themselves can be used to retrain the dialogue agent. In the asymmetric Alexa Prize setting, Shalyminov et al. (2018) show that conversation-level metrics like rating and length can also be used to train an effective ranker.

6.10 Limitations

Our findings on user behavior are particular to the demographics of the US Alexa customers who spoke to CHIRPY in 2019–2020. While users in other locations or time periods may differ, our analysis gives a valuable snapshot of the current attitudes and expectations of US users interacting with a voice-based socialbot or virtual assistant.

Second, our results are dependent on the Alexa Prize conversational context and the technical details of our generative model. In particular, due to latency and cost constraints, our GPT2-medium generative model is orders of magnitude smaller than the current largest generative models, and trained on a fraction of the data (Brown et al., 2020; Adiwardana et al., 2020). Given that very large models have shown generative abilities that are absent at smaller scale, it is likely that if we had built our dialogue agent with such a model, its errors and interactions with users would have been very different. Nonetheless, we believe our analysis gives useful insight into the performance of neural generative models of more accessible scale, in particular highlighting issues occurring in real-life scenarios that might not occur in crowdsourced conversations.

6.11 Conclusion

As shown by the positive results from recent large pretrained models (Zhang et al., 2020b; Adiwardana et al., 2020; Roller et al., 2021), neural generation can enable more powerful social chatbots, capable of flexibly discussing a much greater range of topics than previous rule-based or retrieval-based systems. Indeed, when building CHIRPY, a modular open-domain socialbot, we found that neural generation is a valuable addition that improves CHIRPY's flexibility and conversational tone (Paranjape et al., 2020).

However, this chapter shows that in this real-life setting, poor-quality bot turns are common. The noisy environment – in which user utterances are often unclear – plays a large part in the bot's more basic errors (repetition, ignoring, and nonsensical utterances). However, even in clear examples where the generative model could succeed, it still makes many unforced errors; these are more likely to involve faults in reasoning or social abilities. This highlights the importance of improving neural generative dialogue models' state-tracking, commonsense abilities and use of conversational history.

Despite the frequency of errors, users are generally polite; most don't express overt dissatisfaction even after an error. However, *unaddressed* dissatisfaction escalates: it makes users more critical, offensive, and likely to quit when encountering an error, and more offensive even if there are no further errors. We find that dissatisfaction correlates with bot errors, however, it can arise unpredictably for other reasons – e.g., as a result of privacy boundaries, which are variable and personal to each user.

Dissatisfaction is relatively easy to automatically *detect*, and thus feasible as a scalable semi-supervised learning signal that could be used for online learning. However, it is difficult to *predict*; this makes it a challenging learning signal. Indeed, we find that our predictor conditions more strongly on easier-to-recognize factors such as prior user dissatisfaction, than on harder-to-recognize factors such as bot errors. Nonetheless, we find that when used as a ranking function to choose between alternative bot utterances, the predictor chooses better than random selection.

Chapter 7

Conclusion

Open-ended text generation has the potential to enable AI systems to communicate complex information to humans, in the form best understood by us – natural language. In this thesis, I have focused on the transformative advances achieved by applying Deep Learning techniques to this task. I have sought to understand and improve upon the main problems in these systems, and by observing their impact on user experience, to assess their suitability for real-life deployment.

In Chapter 3, we identified two main problems in a neural abstractive text summarization model – copying accuracy and repetition. We showed that applying a hybrid pointer-generator system and a coverage mechanism can improve on these problems in the context of abstractive text summarization. In Chapter 4, we identified several behavioral problems in a neural generative chitchat model. Through a detailed multi-turn human evaluation, we identified how these problems affect different aspects of user experience. We showed that certain controlled text generation methods can be effective to manipulate these behaviors; thus improving user experience. In Chapter 5, we compared an extensively pretrained Language Model to one that was not, on the task of narrative text generation. By evaluating several syntactic, semantic, structural, and stylistic aspects of the generated text, we characterized the text generated by these systems. In particular, by evaluating the generated text across the whole decoding algorithm spectrum, we emphasized the often-overlooked impact of decoding algorithms. In Chapter 6, we presented our findings from putting a neural generative chitchat model in deployment talking to real, intrinsically-motivated users.

We use this rare opportunity to characterize the types of errors made by the bot, and how they affect user dissatisfaction. Though we find that our neural generative system is still far from a reliable user experience, we demonstrate a semi-supervised method to learn from user dissatisfaction, and thus improve the bot.

Promise and risks of open-ended text generation. Looking to the future, open-ended neural text generation presents both substantial promise and risks. On the one hand, knowledgeable dialogue systems could enable an accessible and socially-aware method for knowledge dissemination; this could be vital in an age of misinformation and echo chambers. More generally, greater text generation abilities mean a more natural interface to computers, which could broaden computing access to people of different backgrounds and skills. Open-ended text generation also offers many exciting creative opportunities, opening a new toolbox to artists, writers and game developers; it could even inspire new genres of art.

On the other hand, more convincing open-ended text generation has considerable scope for misuse, for example generating fake news or online harassment. They could also perpetuate many types of unintentional harms, in particular various manifestations of bias as described in Section 1.3.3. The personification of dialogue agents, and our interactions with them, can enforce troubling societal power structures – for example, the prevailing female gendering of virtual assistants carries associations of female submissiveness. Furthermore, while more socially engaging machines could enable users to form meaningful relationships with them, they could worsen our existing crises of digital addiction, isolation and mental health problems. Together, these opportunities and dangers intersect with many of the most pressing social issues of our era.

Future directions. As I argued in Section 1.6, I believe that the related problems of control and safety are the most important to solve, if we wish to realize the benefits and minimize the harms described above. As larger pretrained LMs make open-ended neural text generation more of a practical possibility, I'm glad to see the research community increasingly focused on this area.

Though not explored in this thesis, grounded and multi-modal text generation is an important future direction, particularly for dialogue models. In Chapter 6, we found that the user's tone of voice and body language were very important missing pieces in the social conversation. Incorporating this information – for both the user and the bot – will be essential to truly understand and convey social context. Like many other goals, this will only be possible through interdisciplinary collaboration with fields such as Human-Computer Interaction.

The value of research. In this thesis I have endeavored to provide an overview of open-ended neural text generation at this point in time. But given how rapidly the field has changed in the last few years, it's natural to wonder: how much of this snapshot will be relevant a few years from now?

I believe that the value of research lies not only in whether a particular technique is used in the future – in fact, almost all research will eventually fail that criterion! The value of research also lies in *aiding the collective understanding* – using open science to bring a wider view to more people, as I described in Section 1.6. Indeed, throughout my PhD I have often found this kind of contribution the most rewarding. I hope this thesis will inspire the reader to contribute to the collective understanding, which will be passed from generation to generation of researchers.

Lastly, the history of Deep Learning shows that some good ideas – such as the foundational work of the 80s and 90s – may need to wait for the right conditions to flourish. In Section 1.6 I described the wave of large pretrained LMs as a homogenization of the field, subsuming other techniques. When the tide eventually goes out, those other techniques will still be there, and they may have something new to offer.

Bibliography

Mohsen Abbasi, Sorelle A Friedler, Carlos Scheidegger, and Suresh Venkatasubramanian. 2019. Fairness in representation: quantifying stereotyping as a representational harm. In *SIAM International Conference on Data Mining*, pages 801–809.

Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. 2020. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*.

Ali Araabi and Christof Monz. 2020. Optimizing transformer for low-resource neural machine translation. In *International Conference on Computational Linguistics (COLING)*, pages 3429–3435.

Shlomo Argamon, Moshe Koppel, and Galit Avneri. 1998. Routing documents according to style. In *International workshop on innovative information systems*.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations (ICLR)*.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations (ICLR)*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations (ICLR)*.

Ashutosh Baheti, Alan Ritter, Jiwei Li, and Bill Dolan. 2018. Generating more interesting responses in neural conversation models with distributional constraints. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Regina Barzilay and Mirella Lapata. 2008. Modeling local coherence: An entity-based approach. *Computational Linguistics*, 34(1):1–34.

Scott F Beers and William E Nagy. 2009. Syntactic complexity as a predictor of adolescent writing quality: Which measures? Which genre? *Reading and Writing*, 22(2):185–200.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623.

Emily M Bender and Alexander Koller. 2020. Climbing towards NLU: On meaning, form, and understanding in the age of data. In *Association for Computational Linguistics (ACL)*, pages 5185–5198.

Yoshua Bengio, Patrice Simard, and Paolo Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. 2020. Language (technology) is power: A critical survey of 'bias' in NLP. In *Association for Computational Linguistics (ACL)*, pages 5454–5476.

Marcely Zanon Boito, Aline Villavicencio, and Laurent Besacier. 2019. Empirical evaluation of sequence-to-sequence models for word discovery in low-resource settings. *arXiv preprint arXiv:1907.00184*.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby

Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Kohd, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. 2021. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*.

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *International Conference on Learning Representations (ICLR)*.

Shikha Bordia and Samuel Bowman. 2019. Identifying and reducing gender bias in word-level language models. In *NAACL Student Research Workshop*, pages 7–15.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Computational Natural Language Learning (CoNLL)*, pages 10–21.

Margarita Boyarskaya, Alexandra Olteanu, and Kate Crawford. 2020. Overcoming failures of imagination in AI infused system development and deployment. In *NeurIPS Workshop on Navigating the Broader Impacts of AI Research*.

Denny Britz, Anna Goldie, Minh-Thang Luong, and Quoc Le. 2017. Massive exploration of neural machine translation architectures. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1442–1451.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Marc Brysbaert, Amy Beth Warriner, and Victor Kuperman. 2014. Concreteness ratings for 40 thousand generally known English word lemmas. *Behavior Research Methods*, 46(3):904–911.

Massimo Caccia, Lucas Caccia, William Fedus, Hugo Larochelle, Joelle Pineau, and Laurent Charlin. 2018. Language GANs falling short. In *NeurIPS Workshop on Critiquing and Correcting Trends in Machine Learning*.

Ziqiang Cao, Furu Wei, Wenjie Li, and Sujian Li. 2018. Faithful to the original: Fact aware neural abstractive summarization. In *Conference on Artificial Intelligence (AAAI)*, volume 32.

Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, et al. 2020. Extracting training data from large language models. *arXiv preprint arXiv:2012.07805*.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harri Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, et al. 2018. The best of both worlds: Combining recent advances in neural machine translation. In *Association for Computational Linguistics (ACL)*, pages 76–86.

Mia Xu Chen, Benjamin N Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M Dai, Zhifeng Chen, et al. 2019. Gmail smart compose:

Real-time assisted writing. In *ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2287–2295.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. 2016. Distraction-based neural networks for modeling documents. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Jackie Chi Kit Cheung and Gerald Penn. 2014. Unsupervised sentence enhancement for automatic summarization. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Ethan A. Chi, Caleb Chiam, Trenton Chang, Swee Kiat Lim, Chetanya Rastogi, Alexander Iyabor, Yutong He, Hari Sowrirajan, Avanika Narayan, Jillian Tang, Haojun Li, Ashwin Paranjape, and Christopher D. Manning. 2021. Neural, neural everywhere: Controlled generation meets scaffolded, structured dialogue. In *Alexa Prize Proceedings*.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Sumit Chopra, Michael Auli, and Alexander M Rush. 2016. Abstractive sentence summarization with attentive recurrent neural networks. In *North American Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah A. Smith. 2021. All that's 'human' is not gold: Evaluating human evaluation of generated text. In *Association for Computational Linguistics (ACL)*, pages 7282–7296.

Arman Cohan, Franck Dernoncourt, Doo Soon Kim, Trung Bui, Seokhwan Kim, Walter Chang, and Nazli Goharian. 2018. A discourse-aware attention model for abstractive summarization of long documents. In *North American Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 615–621.

Eldan Cohen and Christopher Beck. 2019. Empirical analysis of beam search performance degradation in neural sequence models. In *International Conference on Machine Learning (ICML)*, volume 97, pages 1290–1299.

Kate Crawford. 2017. The trouble with bias. Keynote at NeurIPS.

Emmelyn AJ Croes and Marjolijn L Antheunis. 2020. 36 questions to loving a chatbot: Are people willing to self-disclose to a chatbot? In *International Workshop on Chatbot Research and Design*, pages 81–95.

Robert Dale, Barbara Di Eugenio, and Donia Scott. 1998. Introduction to the special issue on natural language generation. *Computational Linguistics*, 24(3):345–353.

Antonella De Angeli and Sheryl Brahnam. 2008. I hate you! Disinhibition with virtual partners. *Interacting with computers*, 20(3):302–310.

Antonella De Angeli, Rollo Carpenter, et al. 2005. Stupid computer! Abuse and social identities. In *Abuse: The darker side of Human-Computer Interaction (INTERACT Workshop)*, pages 19–25.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *EACL Workshop on Statistical Machine Translation*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *North American Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Emily Dinan, Angela Fan, Adina Williams, Jack Urbanek, Douwe Kiela, and Jason Weston. 2020. Queens are powerful too: Mitigating gender bias in dialogue generation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 8173–8188.

Emily Dinan, Varvara Logacheva, Valentin Malykh, Alexander Miller, Kurt Shuster, Jack Urbanek, Douwe Kiela, Arthur Szlam, Iulian Serban, Ryan Lowe, et al. 2019a. The second conversational intelligence challenge (ConvAI2). *arXiv preprint arXiv:1902.00098*.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019b. Wizard of Wikipedia: Knowledge-powered conversational agents. In *International Conference on Learning Representations (ICLR)*.

Kathryn Dindia. 2002. Self-disclosure research: Knowledge through meta-analysis. *Interpersonal communication research: Advances through meta-analysis*, pages 169–185.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *The Journal of Machine Learning Research (JMLR)*, 12:2121–2159.

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: a corpus for adding memory to goal-oriented dialogue systems. In *Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 207–219.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Alexander R Fabbri, Wojciech Kryscinski, Bryan McCann, Caiming Xiong, Richard Socher, and Dragomir Radev. 2021. SummEval: Re-evaluating summarization evaluation. *Transactions of the Association of Computational Linguistics (TACL)*, 9:391–409.

Angela Fan, David Grangier, and Michael Auli. 2018a. Controllable abstractive summarization. In *Workshop on Neural Machine Translation and Generation (WNMT)*, pages 45–54.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018b. Hierarchical neural story generation. In *Association for Computational Linguistics (ACL)*.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Workshop on Stylistic Variation*, pages 94–104.

Marina Fomicheva and Lucia Specia. 2019. Taking MT evaluation metrics to extremes: Beyond correlation with human judgments. *Computational Linguistics*, 45(3):515–558.

Markus Freitag, George Foster, David Grangier, Viresh Ratnakar, Qijun Tan, and Wolfgang Macherey. 2021. Experts, errors, and context: A large-scale study of human evaluation for machine translation. *arXiv preprint arXiv:2104.14478*.

Markus Freitag, David Grangier, and Isaac Caswell. 2020. BLEU might be guilty but references are not innocent. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 61–71.

Raefer Gabriel, Yang Liu, Anna Gottardi, Mihail Eric, Anju Khatri, Anjali Chadha, Qinlang Chen, Behnam Hedayatnia, Pankaj Rajan, Ali Binici, et al. 2020. Further advances in open domain dialog systems in the third Alexa Prize socialbot grand challenge. In *Alexa Prize Proceedings*.

Xiang Gao, Yizhe Zhang, Michel Galley, Chris Brockett, and William B Dolan. 2020. Dialogue response ranking training with large-scale human feedback data. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 386–395.

Ismael Garrido-Muñoz, Arturo Montejo-Ráez, Fernando Martínez-Santiago, and L Alfonso Ureña-López. 2021. A survey on bias in deep NLP. *Applied Sciences*, 11(7):3184.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. 2020. RealToxicityPrompts: Evaluating neural toxic degeneration in language models. In *Empirical Methods in Natural Language Processing (EMNLP): Findings*, pages 3356–3369.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *International Conference on Machine Learning (ICML)*.

Sebastian Gehrmann, Hendrik Strobelt, and Alexander Rush. 2019. GLTR: Statistical detection and visualization of generated text. In *Association for Computational Linguistics (ACL): System Demonstrations*, pages 111–116.

Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural computation*, 12(10):2451–2471.

Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. 2017. Hafez: an interactive poetry generation system. In *Association for Computational Linguistics (ACL): System Demonstrations*, pages 43–48.

Julia Gong. 2018. Google Translate's 'sinister religious prophecies', demystified. *Skynet Today blog post*.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. *Advances in Neural Information Processing Systems (NeurIPS)*, 27.

Ben Green. 2019. 'Good' isn't good enough. In *NeurIPS AI for Social Good workshop*.

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

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Association for Computational Linguistics (ACL)*.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. In *Association for Computational Linguistics (ACL)*.

Fenfei Guo, Angeliki Metallinou, Chandra Khatri, Anirudh Raju, Anu Venkatesh, and Ashwin Ram. 2018. Topic-based evaluation for conversational bots. *NeurIPS Conversational AI Workshop*.

Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazare, and Jason Weston. 2019. Learning from dialogue after deployment: Feed yourself, chatbot! In *Association for Computational Linguistics (ACL)*, pages 3667–3684.

Amelia Hardy, Ashwin Paranjape, and Christopher Manning. 2021. Effective social chatbot strategies for increasing user initiative. In *Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 99–110.

Tatsunori Hashimoto, Hugh Zhang, and Percy Liang. 2019. Unifying human and statistical evaluation for natural language generation. In *North American Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Helen Hastie. 2012. Metrics and evaluation of spoken dialogue systems. In *Data-driven methods for adaptive spoken dialogue systems*, pages 131–150. Springer.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. 2018. Learning to write with cooperative discriminators. In *Association for Computational Linguistics (ACL)*, pages 1638–1649.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations (ICLR)*.

Shui Hu, Yang Liu, Anna Gottardi, Behnam Hedayatnia, Anju Khatri, Anjali Chadha, Qinlang Chen, Pankaj Rajan, Ali Binici, Varun Somani, Yao Lu, Prerna Dwivedi, Lucy Hu, Hangjie Shi, Sattvik Sahai, Mihail Eric, Karthik Gopalakrishnan, Seokhwan Kim, Spandana Gella, Alexandros Papangelis, Patrick Lange, Di Jin, Nicole Chartier, Mahdi Namazifar, Aishwarya Padmakumar, Sarik Ghazarian, Shereen Oraby, Anjali Narayan-Chen, Yuheng Du, Lauren Stubell, Savanna Stiff, Kate Bland, Arindam Mandal, Reza Ghanadan, and Dilek Hakkani-Tur. 2021. Further advances in open domain dialog systems in the fourth Alexa Prize socialbot grand challenge. In *Alexa Prize Proceedings*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *International Conference on Machine Learning (ICML)*.

Po-Sen Huang, Huan Zhang, Ray Jiang, Robert Stanforth, Johannes Welbl, Jack Rae, Vishal Maini, Dani Yogatama, and Pushmeet Kohli. 2020. Reducing sentiment bias in language models via counterfactual evaluation. In *Empirical Methods in Natural Language Processing (EMNLP): Findings*, pages 65–83.

Touseef Iqbal and Shaima Qureshi. 2020. The survey: Text generation models in deep learning. *Journal of King Saud University – Computer and Information Sciences*.

Molly E Ireland and James W Pennebaker. 2010. Language style matching in writing: Synchrony in essays, correspondence, and poetry. *Journal of personality and social psychology*, 99(3):549.

Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. 2016. Summarizing source code using a neural attention model. In *Association for Computational Linguistics (ACL)*, pages 2073–2083.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. 2019. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*.

Robin Jia and Percy Liang. 2016. Data recombination for neural semantic parsing. In *Association for Computational Linguistics (ACL)*, pages 12–22.

Shaojie Jiang and Maarten de Rijke. 2018. Why are sequence-to-sequence models so dull? Understanding the low-diversity problem of chatbots. In *EMNLP Workshop on Search-Oriented Conversational AI*.

Hongyan Jing. 2000. Sentence reduction for automatic text summarization. In *Applied natural language processing*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2017. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734*.

Daniel Jurafsky and James H. Martin. 2009. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA.

Yaser Keneshloo, Tian Shi, Naren Ramakrishnan, and Chandan K Reddy. 2019. Deep reinforcement learning for sequence-to-sequence models. *IEEE transactions on neural networks and learning systems*, 31(7):2469–2489.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

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

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, Fog count and Flesch reading ease formula) for navy enlisted personnel. Technical report, Naval Technical Training Command Millington TN Research Branch.

Diederik P Kingma and Max Welling. 2014. Auto-encoding variational Bayes. *stat*, 1050:1.

Philipp Koehn. 2009. *Statistical machine translation*. Cambridge University Press.

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. In *Workshop on Neural Machine Translation and Generation (WNMT)*, pages 28–39.

Anastassia Kornilova and Vladimir Eidelman. 2019. BillSum: A corpus for automatic summarization of US legislation. In *Workshop on New Frontiers in Summarization*, pages 48–56.

Wojciech Kryściński, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Neural text summarization: A critical evaluation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 540–551.

Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. 2020. Evaluating the factual consistency of abstractive text summarization. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346.

Ilia Kulikov, Alexander Miller, Kyunghyun Cho, and Jason Weston. 2019. Importance of search and evaluation strategies in neural dialogue modeling. In *International Conference on Natural Language Generation (ICNLG)*, pages 76–87.

Julian Kupiec, Jan Pedersen, and Francine Chen. 1995. A trainable document summarizer. In *International ACM SIGIR conference on Research and development in information retrieval*.

Andrey Kurenkov and Arnav Arora. 2019. OpenAI's GPT2 - food to media hype or wake up call? *Skynet Today blog post*.

Alex M Lamb, Anirudh Goyal Alias Parth Goyal, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4601–4609.

Samuel Läubli, Sheila Castilho, Graham Neubig, Rico Sennrich, Qinlan Shen, and Antonio Toral. 2020. A set of recommendations for assessing human–machine parity in language translation. *Journal of Artificial Intelligence Research*, 67:653–672.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016a. A diversity-promoting objective function for neural conversation models. In *North American Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Jiwei Li and Dan Jurafsky. 2016. Mutual information and diverse decoding improve neural machine translation. *arXiv preprint arXiv:1601.00372*.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2017a. Learning to decode for future success. *arXiv preprint arXiv:1701.06549*.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016b. Deep reinforcement learning for dialogue generation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1192–1202.

Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017b. Adversarial learning for neural dialogue generation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2157–2169.

Margaret Li, Jason Weston, and Stephen Roller. 2019. ACUTE-EVAL: Improved dialogue evaluation with optimized questions and multi-turn comparisons. *arXiv preprint arXiv:1909.03087*.

Chin-Yew Lin. 2004a. Looking for a few good metrics: Automatic summarization evaluation – how many samples are enough? In *NACSIS/NII Test Collection for Information Retrieval (NTCIR) Workshop*.

Chin-Yew Lin. 2004b. ROUGE: A package for automatic evaluation of summaries. In *Text summarization branches out (ACL workshop)*.

Wang Ling, Chris Dyer, Alan W Black, Isabel Trancoso, Ramón Fermandez, Silvio Amir, Luis Marujo, and Tiago Luís. 2015. Finding function in form: Compositional character models for open vocabulary word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1520–1530.

Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Haochen Liu, Jamell Dacon, Wenqi Fan, Hui Liu, Zitao Liu, and Jiliang Tang. 2020. Does gender matter? Towards fairness in dialogue systems. In *International Conference on Computational Linguistics (COLING)*, pages 4403–4416.

Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. 2017. Towards an automatic Turing test: Learning to evaluate dialogue responses. In *Association for Computational Linguistics (ACL)*, pages 1116–1126.

Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. 2015. The Ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. In *Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 285–294.

Minh-Thang Luong and Christopher D Manning. 2016. Achieving open vocabulary neural machine translation with hybrid word-character models. In *Association for Computational Linguistics (ACL)*, pages 1054–1063.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Qingsong Ma, Johnny Wei, Ondřej Bojar, and Yvette Graham. 2019. Results of the WMT19 metrics shared task: Segment-level and strong MT systems pose big challenges. In *Conference on Machine Translation (WMT)*, pages 62–90.

Yukun Ma, Khanh Linh Nguyen, Frank Z Xing, and Erik Cambria. 2020. A survey on empathetic dialogue systems. *Information Fusion*, 64:50–70.

Christopher D. Manning and Abigail See. 2019. CS224n: Natural language processing with deep learning. Lecture slides.

Huanru Henry Mao, Bodhisattwa Prasad Majumder, Julian McAuley, and Garrison W. Cottrell. 2019. Improving neural story generation by targeted common sense grounding. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Benjamin Marie, Atsushi Fujita, and Raphael Rubino. 2021. Scientific credibility of machine translation research: A meta-evaluation of 769 papers. In *Association for Computational Linguistics (ACL)*.

Luca Massarelli, Fabio Petroni, Aleksandra Piktus, Myle Ott, Tim Rocktäschel, Vassilis Plachouras, Fabrizio Silvestri, and Sebastian Riedel. 2020. How decoding strategies affect the verifiability of generated text. In *Empirical Methods in Natural Language Processing (EMNLP): Findings*, pages 223–235.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Association for Computational Linguistics (ACL)*, pages 1906–1919.

Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2018. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*.

Danielle S McNamara, Scott A Crossley, and Philip M McCarthy. 2010. Linguistic features of writing quality. *Written communication*, 27(1):57–86.

Clara Meister, Ryan Cotterell, and Tim Vieira. 2020. If beam search is the answer, what was the question? In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2173–2185.

Gaurav Menghani. 2021. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *arXiv preprint arXiv:2106.08962*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. In *NeurIPS Workshop on Multi-class and Multi-label Learning in Extremely Large Label Spaces*.

Haitao Mi, Baskaran Sankaran, Zhiguo Wang, and Abe Ittycheriah. 2016. Coverage embedding models for neural machine translation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Julian Michael. 2020. To dissect an octopus: Making sense of the form/meaning debate. *Blog post*.

Sabrina J Mielke, Arthur Szlam, Y-Lan Boureau, and Emily Dinan. 2020. Linguistic calibration through metacognition: Aligning dialogue agent responses with expected correctness. *arXiv preprint arXiv:2012.14983*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *International Conference on Learning Representations (ICLR), Workshop Track Proceedings*.

Alexander Miller, Will Feng, Dhruv Batra, Antoine Bordes, Adam Fisch, Jiasen Lu, Devi Parikh, and Jason Weston. 2017. ParlAI: A dialog research software platform. In *Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*, pages 79–84.

Lili Mou and Olga Vechtomova. 2020. Stylized text generation: Approaches and applications. In *Association for Computational Linguistics (ACL)*. Tutorial presentation.

Moin Nadeem, Anna Bethke, and Siva Reddy. 2020. StereoSet: Measuring stereotypical bias in pretrained language models. *arXiv preprint arXiv:2004.09456*.

Sathvik Nair and Daniel Bashir. 2020. GPT-3: An AI breakthrough, but not coming for your job. *Skynet Today blog post*.

Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. 2017. SummaRuNNer: A recurrent neural network based sequence model for extractive summarization of documents. In *Conference on Artificial Intelligence (AAAI)*.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çaglar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence RNNs and beyond. In *Computational Natural Language Learning (CoNLL)*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1797–1807.

Ani Nenkova and Kathleen McKeown. 2011. Automatic summarization. *Information Retrieval*, 5(2-3):103–233.

Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why we need new evaluation metrics for NLG. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Myle Ott, Michael Auli, David Grangier, and Marc'Aurelio Ranzato. 2018. Analyzing uncertainty in neural machine translation. In *International Conference on Machine Learning (ICML)*, volume 80, pages 3956–3965.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. FAIRSEQ: A fast, extensible toolkit for sequence modeling. In *North American Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Artidoro Pagnoni, Vidhisha Balachandran, and Yulia Tsvetkov. 2021. Understanding factuality in abstractive summarization with FRANK: A benchmark for factuality metrics. In *North American Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4812–4829.

Chris D Paice. 1990. Constructing literature abstracts by computer: techniques and prospects. *Information Processing & Management*, 26(1):171–186.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Association for Computational Linguistics (ACL)*, pages 311–318.

Ashwin Paranjape, Abigail See, Kathleen Kenealy, Haojun Li, Amelia Hardy, Peng Qi, Kaushik Ram Sadagopan, Nguyet Minh Phu, Dilara Soylu, and Christopher D Manning. 2020. Neural generation meets real people: Towards emotionally engaging mixed-initiative conversations. In *Alexa Prize Proceedings*.

Prasanna Parthasarathi and Joelle Pineau. 2018. Extending neural generative conversational model using external knowledge sources. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 690–695.

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

Xiangyu Peng, Siyan Li, Spencer Frazier, and Mark Riedl. 2020. Reducing non-normative text generation from language models. In *International Conference on Natural Language Generation (ICNLG)*, pages 374–383.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *North American Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Horst Pöttker. 2003. News and its communicative quality: the inverted pyramid—when and why did it appear? *Journalism Studies*, 4(4):501–511.

Shrimai Prabhumoye, Elijah Mayfield, and Alan W Black. 2019. Principled frameworks for evaluating ethics in NLP systems. In *Workshop on Widening NLP*, pages 118–121.

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

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI tech report*.

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

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations (ICLR)*.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2019. Towards empathetic open-domain conversation models: A new benchmark and dataset. In *Association for Computational Linguistics (ACL)*, pages 5370–5381.

Melissa Roemmele, Andrew S Gordon, and Reid Swanson. 2017. Evaluating story generation systems using automated linguistic analyses. In *KDD Workshop on Machine Learning for Creativity*.

Stephen Roller, Emily Dinan, Naman Goyal, Da Ju, Mary Williamson, Yinhan Liu, Jing Xu, Myle Ott, Eric Michael Smith, Y-Lan Boureau, et al. 2021. Recipes for building an open-domain chatbot. In *European Chapter of the Association for Computational Linguistics (EACL)*, pages 300–325.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Horacio Saggion and Thierry Poibeau. 2013. Automatic text summarization: Past, present and future. In *Multi-source, Multilingual Information Extraction and Summarization*, pages 3–21. Springer.

Nithya Sambasivan, Shivani Kapania, Hannah Highfill, Diana Akrong, Praveen Paritosh, and Lora M Aroyo. 2021. 'Everyone wants to do the model work, not the data work': Data cascades in high-stakes AI. In *CHI Conference on Human Factors in Computing Systems*, pages 1–15.

Chinnadhurai Sankar, Sandeep Subramanian, Christopher Pal, Sarath Chandar, and Yoshua Bengio. 2019. Do neural dialog systems use the conversation history effectively? An empirical study. In *Association for Computational Linguistics (ACL)*, pages 32–37.

Baskaran Sankaran, Haitao Mi, Yaser Al-Onaizan, and Abe Ittycheriah. 2016. Temporal attention model for neural machine translation. *arXiv preprint arXiv:1608.02927*.

Jens Schanze. 2010. Plug & pray. Documentary film.

William A Scott. 1955. Reliability of content analysis: The case of nominal scale coding. *Public opinion quarterly*, pages 321–325.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Association for Computational Linguistics (ACL)*, pages 1073–1083.

Abigail See and Christopher D. Manning. 2021. Understanding and predicting user dissatisfaction in a neural generative chatbot. In *Special Interest Group on Discourse and Dialogue (SIGDIAL)*.

Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, and Christopher D. Manning. 2019a. Do massively pretrained language models make better storytellers? In *Computational Natural Language Learning (CoNLL)*, pages 843–861.

Abigail See, Stephen Roller, Douwe Kiela, and Jason Weston. 2019b. What makes a good conversation? How controllable attributes affect human judgments. In *North American Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Association for Computational Linguistics (ACL)*, pages 7881–7892.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Association for Computational Linguistics (ACL)*, pages 1715–1725.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016a. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Conference on Artificial Intelligence (AAAI)*.

Iulian Vlad Serban, Ryan Lowe, Laurent Charlin, and Joelle Pineau. 2016b. Generative deep neural networks for dialogue: A short review. *NeurIPS Workshop on Learning Methods for Dialogue*.

Deven Santosh Shah, H. Andrew Schwartz, and Dirk Hovy. 2020. Predictive biases in natural language processing models: A conceptual framework and overview. In *Association for Computational Linguistics (ACL)*, pages 5248–5264.

Igor Shalyminov, Ondřej Dušek, and Oliver Lemon. 2018. Neural response ranking for social conversation: A data-efficient approach. In *EMNLP Workshop on Search-Oriented Conversational AI*, pages 1–8.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *Association for Computational Linguistics (ACL)*.

Eva Sharma, Chen Li, and Lu Wang. 2019. BIGPATENT: A large-scale dataset for abstractive and coherent summarization. In *Association for Computational Linguistics (ACL)*, pages 2204–2213.

Xiaoyu Shen, Hui Su, Yanran Li, Wenjie Li, Shuzi Niu, Yang Zhao, Akiko Aizawa, and Guoping Long. 2017. A conditional variational framework for dialog generation. In *Association for Computational Linguistics (ACL)*, pages 504–509.

Emily Sheng, Kai-Wei Chang, Prem Natarajan, and Nanyun Peng. 2019. The woman worked as a babysitter: On biases in language generation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 3407–3412.

Emily Sheng, Kai-Wei Chang, Prem Natarajan, and Nanyun Peng. 2020. Towards controllable biases in language generation. In *Empirical Methods in Natural Language Processing (EMNLP): Findings*, pages 3239–3254.

Jamin Shin, Peng Xu, Andrea Madotto, and Pascale Fung. 2019. HappyBot: Generating empathetic dialogue responses by improving user experience look-ahead. *arXiv preprint arXiv:1906.08487*.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. *arXiv preprint arXiv:2104.07567*.

Kurt Shuster, Jack Urbanek, Emily Dinan, Arthur Szlam, and Jason Weston. 2020. Deploying lifelong open-domain dialogue learning. *arXiv preprint arXiv:2008.08076*.

Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. 2018. Cold fusion: Training seq2seq models together with language models. In *Interspeech*, pages 387–391.

Artūrs Stafanovičs, Mārcis Pinnis, and Toms Bergmanis. 2020. Mitigating gender bias in machine translation with target gender annotations. In *Conference on Machine Translation (WMT)*, pages 629–638.

Felix Stahlberg and Bill Byrne. 2019. On NMT search errors and model errors: Cat got your tongue? In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 3356–3362.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Association for Computational Linguistics (ACL)*, pages 3645–3650.

Tony Sun, Andrew Gaut, Shirlyn Tang, Yuxin Huang, Mai ElSherief, Jieyu Zhao, Diba Mirza, Elizabeth Belding, Kai-Wei Chang, and William Yang Wang. 2019. Mitigating gender bias in natural language processing: Literature review. In *Association for Computational Linguistics (ACL)*, pages 1630–1640.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 3104–3112.

Jun Suzuki and Masaaki Nagata. 2017. Cutting-off redundant repeating generations for neural abstractive summarization. In *European Chapter of the Association for Computational Linguistics (EACL)*, pages 291–297.

Ayesha Ayub Syed, Ford Lumban Gaol, and Tokuro Matsuo. 2021. A survey of the state-of-the-art models in neural abstractive text summarization. *IEEE Access*, 9:13248–13265.

Sho Takase, Jun Suzuki, Naoaki Okazaki, Tsutomu Hirao, and Masaaki Nagata. 2016. Neural headline generation on abstract meaning representation. In *Empirical Methods in Natural Language Processing (EMNLP)*.

Bowen Tan, Zichao Yang, Maruan Al-Shedivat, Eric Xing, and Zhiting Hu. 2021. Progressive generation of long text with pretrained language models. In *North American Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 4313–4324.

Christopher Taylor. 1998. *Language to language: A practical and theoretical guide for Italian/English translators*. Cambridge University Press.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Association for Computational Linguistics (ACL)*.

Chris Van Der Lee, Albert Gatt, Emiel Van Miltenburg, Sander Wubben, and Emiel Krahmer. 2019. Best practices for the human evaluation of automatically generated text. In *International Conference on Natural Language Generation (ICNLG)*, pages 355–368.

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

Anu Venkatesh, Chandra Khatri, Ashwin Ram, Fenfei Guo, Raefer Gabriel, Ashish Nagar, Rohit Prasad, Ming Cheng, Behnam Hedayatnia, Angeliki Metallinou, et al. 2017. On evaluating and comparing conversational agents. *NeurIPS Conversational AI Workshop*.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search: Decoding diverse solutions from neural sequence models. In *Conference on Artificial Intelligence (AAAI)*.

James Vincent. 2019. Google's head of translation on fighting bias in language and why AI loves religious texts. *The Verge article*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015a. Pointer networks. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015b. Grammar as a foreign language. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 2773–2781.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. In *ICML Deep Learning Workshop*.

Marilyn A. Walker, Diane J. Litman, Candace A. Kamm, and Alicia Abella. 1997. PARADISE: A framework for evaluating spoken dialogue agents. In *Association for Computational Linguistics (ACL)*, pages 271–280.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing NLP. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2153–2162.

Alex Wang and Kyunghyun Cho. 2019. BERT has a mouth, and it must speak: BERT as a Markov Random Field language model. In *Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, pages 30–36.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2019. SuperGLUE: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP (EMNLP Workshop)*, pages 353–355.

Di Wang, Nebojsa Jojic, Chris Brockett, and Eric Nyberg. 2017. Steering output style and topic in neural response generation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 2140–2150.

Joseph Weizenbaum. 1966. ELIZA – a computer program for the study of natural language communication between man and machine. *Communications of the ACM*, 9(1):36–45.

Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. 2020. Neural text generation with unlikelihood training. In *International Conference on Learning Representations (ICLR)*.

Sean Welleck, Jason Weston, Arthur Szlam, and Kyunghyun Cho. 2019. Dialogue natural language inference. In *Association for Computational Linguistics (ACL)*, pages 3731–3741.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *European Chapter of the Association for Computational Linguistics (EACL)*, pages 438–449.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Empirical Methods in Natural Language Processing (EMNLP): System Demonstrations*, pages 38–45.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Chen Xing, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Conference on Artificial Intelligence (AAAI)*, volume 17, pages 3351–3357.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning (ICML)*, pages 2048–2057.

Xinnuo Xu, Ondřej Dušek, Ioannis Konstas, and Verena Rieser. 2018. Better conversations by modeling, filtering, and optimizing for coherence and diversity. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 3981–3991.

Kaisheng Yao and Geoffrey Zweig. 2015. Sequence-to-sequence neural net models for grapheme-to-phoneme conversion. In *Annual Conference of the International Speech Communication Association*.

Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. 2016. Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382*.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020a. PEGASUS: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning (ICML)*, pages 11328–11339.

Ruqing Zhang, Jiafeng Guo, Yixing Fan, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2018a. Learning to control the specificity in neural response generation. In *Association for Computational Linguistics (ACL)*.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018b. Personalizing dialogue agents: I have a dog, do you have pets too? In *Association for Computational Linguistics (ACL)*, pages 2204–2213.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and William B Dolan. 2020b. DialoGPT: Large-scale generative pre-training for conversational response generation. In *Association for Computational Linguistics (ACL): System Demonstrations*, pages 270–278.

Yuhao Zhang, Derek Merck, Emily Tsai, Christopher D Manning, and Curtis Langlotz. 2020c. Optimizing the factual correctness of a summary: A study of summarizing radiology reports. In *Association for Computational Linguistics (ACL)*.

Ganbin Zhou, Ping Luo, Rongyu Cao, Fen Lin, Bo Chen, and Qing He. 2017. Mechanism-aware neural machine for dialogue response generation. In *Conference on Artificial Intelligence (AAAI)*, pages 3400–3407.

Chenguang Zhu, William Hinthorn, Ruochen Xu, Qingkai Zeng, Michael Zeng, Xuedong Huang, and Meng Jiang. 2021. Enhancing factual consistency of abstractive summarization. In *North American Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 718–733.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *International Conference on Computer Vision (ICCV)*, pages 19–27.

Zachary M Ziegler, Luke Melas-Kyriazi, Sebastian Gehrmann, and Alexander M Rush. 2019. Encoder-agnostic adaptation for conditional language generation. *arXiv preprint arXiv:1908.06938*.