Understanding the Limitations of Using
Large Language Models for Text Generation

Daphne Ippolito

A DISSERTATION

in

Computer and Information Science

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2021

Supervisor of Dissertation

Chris Callison-Burch, Professor, Computer and Information Science

Graduate Group Chairperson

Mayur Naik, Professor of Computer and Information Science

Dissertation Committee

Lyle Ungar, Professor, Computer and Information Science

Dan Roth, Professor, Computer and Information Science

Marianna Apidianaki, Professor, Computer and Information Science

David Grangier, Research Scientist, Google Brain

Understanding the Limitations of Using

Large Language Models for Text Generation

ACKNOWLEDGEMENT

I would like to thank all collaborators at Penn: Reno Kriz I would like to thank all my collaborators and mentors at Google: Daniel Duckworth, Hugh Zhang. Katherine Lee, Nicholas Carlini, Chiyuan Zhang, Ann Yuan, Emily Reif, Noah Fiedel, and Andy Coenen.

Understanding the Limitations of Using
Large Language Models for Text Generation

ABSTRACT

State-of-the-art neural language models are capable of generating incredibly fluent English text. In my proposed dissertation, I aim to explore how these models can be applied for creative purposes, such as story generation and building creative tools for writers. I also address several challenges around evaluating the use of neural language models for open-ended text generation. I propose an analysis of the tradeoff between generating lexically diverse text and text humans perceive as high quality. I introduce a detection-based evaluation task that can be used to investigate this tradeoff as well as to study quality differences between natural language generation systems. Lastly, I propose an investigation into the extent to which the giant language models used for text generation produce text which verbatim copies from the training set.

CONTENTS

1 INTRODUCTION

One of the oldest yet most elusive promises of AI is computers that can converse with humans, not just via rigidly structured templates and programming languages, but in natural language. This problem is formalized as Natural Language Generation (NLG), the task of writing novel sentences in a human language such as English. Modelling developments over the last half-decade have led to natural language generation systems which are capable of producing incredibly fluent text. These systems have been applied to practical domains such as machine translation and text summarization and simplification, but they have also been applied to more fanciful ones, such as story generation, video games development, and tooling for creating writing.

Story generation and creative writing are a particularly interesting domain to investigate the strengths and weaknesses of natural language generation systems. Unlike in machine translation or summarization, where it is critical that the generated text is factual and stays faithful to the source material, in creative domains, the hallucinations and unusual word choices that are pervasive in modern NLG may be beneficial to a creative writer's process. Indeed, ideation tools, tools which suggest writing topics, are commonplace in creative writing circles. Furthermore, AI-assisted creative writing is a useful testbed for the strengths and limitations of modern NLG systems, one that challenges how controllable these systems are and whether they can be interacted with in useful ways.

Chapter 5 describes my contributions to the field of AI-assisted creative writing. It discusses the importance of introducing controllability into natural language generation systems–providing writers the ability to dictate what kind of text gets generated and decide how it interfaces with what they might have already written. In particular, we introduce methods for efficiently supporting a fill-in-the-blank paradigm, where a writer can insert text into any position of their current text **{TODO: citation}**. We also describe a simple recipe for supporting style-transfer into any user-defined style without the need for costly training data acquisition and test-specific model training. Both these approaches are incorporated into Wordcraft, an AI-augmented text processor that provides several interfaces for writers to get feedback and suggestions from an NLG system. In studies with both amateur and skilled writers, we found Wordcraft to be a valuable assistive tool for creative writing.

Modern natural language generation systems rely on neural language models, neural networks trained on billions of words of text in order to represent human language (Chapter 2). Understanding the limitations of these language models and the nature of the text they are capable of generating is crucial to the ultimate use of these systems in real applications such as Wordcraft.

A significant limitation for creative applications like Wordcraft is the difficulty in generating text that is diverse (containing uncommon and interesting words and phrases) and high quality (as perceived by human readers). Chapter **??** focuses on this challenge and its ramifications to the detectability of generated text. While many approaches have been proposed to, given a language model, try to improve the diversity of its generations without sacrificing quality, we show that none quite achieve this goal [74]. Most often, practitioners choose approaches that err on the side of human-perceived quality at the expense of lexical diversity. This decision leaves subtle signatures in the generated text which make it easy for automatic classifiers to distinguish it from genuine human-written text.

The remainder of Chapter **??** focuses on this detection problem. The proliferation of machine-generated text, especially when it lacks attribution, is of significant concern to the public. As tools for generating text become more powerful and easier to use, they could enable nefarious uses such as fabricated product reviews and social media posts. In Ippolito et al. 2019 [72], we measure the ability of humans as well as automatic systems to detect machine-generated text. Understanding detectability is imperative because it gives us a proxy for how far along generative systems are at fooling humans and whether undesired use of machine-generated text can be mitigated.

One difficulty in studying human ability to detect machine-generated text is that it can be very difficult to collect annotations. Many of the errors NLG systems make are subtle and require closely reading several sentences of text to be able to identify. Common strategies for soliciting human annotators, such as paying crowd workers a fixed dollar amount per annotation, do not tend to yield useful annotations since annotators are not incentivized to spend the extra time to do a close read. In order to be able to study human detection ability at scale, we built the Real or Fake Text game (RoFT), a website that gamifies the task of identifying machine-generated text [47]. The RoFT platform allowed us to collect over 40,000 annotations of whether players could correctly identify when a passage of text transitioned from being human-written to being machine-generated . Chapter 3.5 presents a detailed analysis of the factors we found that most impacted detectability.

Of course, machine-generated text is most undetectable when it looks *exactly* like its training data. Large languages are worryingly capable of memorizing and regurgitating significant amounts of their training data. For example, GPT-3, a popular model that has already been incorporated into several products, when prompted with the first sentence of *Harry Potter* or *Lord of the Rings* will accurately generate the first several paragraphs of each book, This behaviour is especially problematic for the domain of AI-assisted creative writing, as writers using tools such as Wordcraft have the expectation that the generations they are being shown are unique and not plagiarized. Memorization also makes the task of studying the dectability of generated text more challenging. If an NLG system generates Chapter 1 of *Harry Potter*, should this text be labeled as human-written or machine-generated? Chapter 4 focuses on this question of memorization. First, we show how performing thorough deduplication of training data results in models that are less likely to exhibit memorization . Then we conduct experiments showing how observable memorization scales with respect to the number of times a sequence occurs in the training set, the model size, and the length of conditioning prompt [25].

1.1 SUMMARY OF CONTRIBUTIONS

something something something

2 BACKGROUND ON TEXT GENERATION

Automatic text generation has been a goal of computer science researchers since the early days of computing. In recent years, algorithm and statistical approaches have given way to neural language models–neural networks trained to build representations of human language from millions or even billions of documents. This chapter gives a brief overview of how the task of text generation was approached before the ascendency of deep learning and then delves into the details of modern text generation using neural language models.

2.1 BRIEF HISTORY OF TEXT GENERATION

2.2 WHAT IS A LANGUAGE MODEL?

A language model is any model that assigns probabilities to sequences of words. Given a sequence of words $w_1, \dots, w_n$, a langueg model outputs the likelihood $P(w_1, \dots, w_n)$ of this sequence. An ideal language model would have high likelihood to natural-sounding text, like the sentences in this paragraph, and low likelihood to gibberish. Most language models the assumption that the likelihood of a word is dependent only on the words that precede it. Thus, the chain rule applies:

$$P(w_1, \dots, w_n) = P(w_1) \times \dots \times P(w_i | w_1, \dots, w_{i-1}) \times \dots \times P(w_n | w_1, \dots, w_{n-1}) \tag{2.1}$$

Before the transition to neural network-based models, the most common form of language model was an *n*-gram model. Instead of trying to estimate the probability of a word given all preceding words, *n*-gram model make the Markov assumption that the probability of a word is only dependent on a fixed number of preceeding words, referred to as an *n*-grams. For example, using a 2-gram model, we would approximate each factor in Equation 2.1 as

$$P(w_i|w_1, \dots, w_{i-1}) \approx P(w_i|w_{i-2}, w_{i-1}) \tag{2.2}$$

An *n*-gram models can be constructed from a corpus of text by simply counting how many times each word in the text is preceeded by each possible *n*-gram.

There are several disadvantages to this *n*-gram based approach to language modelling. First, *n*-gram models tend to be sparse. If a particular <*n*-gram, word> pair never occurs in the corpus, then the model will assign it a probability of 0. As a result, smoothing techniques are often employed to prevent plausible but novel word sequences from being assigned a probability of zero. Second, the complexity of storing an *n*-gram language model grows exponentially with the choice of *n*. In practice, most *n*-gram models used *n* between 1 and 5, which is insufficient for modelling long-term coherence. Third, *n*-gram models cannot represent words which are not in their vocabulary. Such words are typically replaced with a special out-of-vocabulary identifier. Neural language models, described in the following sections, overcome many of these limitations.

2.3 WHAT IS A NEURAL LANGUAGE MODEL?

Neural network-based language models replace the statistical models described in the previous section with a learned function (the neural network) whose output can be used to predict the likelihood of a word sequence. In contrast to *n*-gram models, neural language models are capable of assigning non-zero probability to sequences never seen in their training corpora, and thus they can be used to model sequences of hundreds or even thousands of words.

One of the key advancements in neural language modeling was the transition from operating on sequences of discrete words to operating on sequences continuous vector representations. The sequence of words $w_1, \ldots, w_n$ is mapped to sequence of embedding vectors $\mathbf{y}_1, \ldots, \mathbf{y}_n$. In early work on neural language modeling, these vector representations were computed separately. Algorithms such as word2vec [115] and GloVe [127] were employed to consturct embedding matrices where each row corresponded to a word in the chosen vocabulary. In today's neural language models, the embedding matrix is typically treated as part of the neural language model, initialized randomly than optimized along with the reset of the network. Let $\mathbf{E}_\theta$ be a learned embedding matrix where each row correspond to the vector represention of one word in the vocabulary.

Typical neural language models emit $\hat{\mathbf{y}}_t$, a predicted embedding for the tth position in the sequence given the previous word embeddings in the sequence. This can be written as

$$\hat{\mathbf{y}}_t = f_\theta(\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) \tag{2.3}$$

where f_θ is the neural network and $\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}$ are the embeddings of the previous tokens in the sequence.

To produce a probability distribution for what the next word should be given the previous words, the predicted embedding $\hat{\mathbf{y}}_t$ is multiplied by the embedding matrix $\mathbf{E}_\theta$ to produce a score for each word in the vocabulary. Then a softmax transformation is used to normalize these scores into a probability distribution. Let Y_t be a random variable representing the vocabulary item predicted for the tth position. We then have:

$$P(Y_t = i|\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) = \frac{\exp(\mathbf{E}\hat{\mathbf{y}}_t[i])}{\sum_j \exp(\mathbf{E}\hat{\mathbf{y}}_t[j])} \tag{2.4}$$

where i and j are indexes into the vocabulary.

The learned weights θ are optimized using a log likelihood loss. More precisely, we can write the training loss for a sequence $\mathbf{y}_1, \ldots, \mathbf{y}_n$ as:

$$\mathcal{L} = -\sum_{t=1}^{n} \log P(Y_t = i^* | \mathbf{y}_{1:t-1}) \tag{2.5}$$

$$= -\sum_{t=1}^{n} \log \frac{\exp(\mathbf{E}_\theta \hat{\mathbf{y}}_t[i^*])}{\sum_j \exp(\mathbf{E}\hat{\mathbf{y}}_t[j])} \tag{2.6}$$

$$= -\sum_{t=1}^{n} \mathbf{E}_\theta \hat{\mathbf{y}}_t[i^*] \tag{2.7}$$

$$= -\sum_{t=1}^{n} (\mathbf{E}_\theta f_\theta(\mathbf{y}_1, \ldots, \mathbf{y}_{t-1})[i^*] \tag{2.8}$$

In these equations, i^* is the index of the groundtruth word at position t in the sequence. By taking the dot product between the neural network's predicted embedding and the embedding of the true word at each position t (Eq. 2.7), we get a score for how correct the neural network's prediction for this position is. Training with an objective of maximizing the sum of these scores over every word position is equivalent to minimizing the negative log likelihood (or maximizing the likelihood) of the sequence.

In some language modelling application, it is common to have an additional sequence which the model is conditioned on in addition to the tokens of the target sequence. This paradigm is known as an encoder-decoder or sequence-to-sequence, and the formulation above is modified to

$$\hat{\mathbf{y}}_t = f_\theta(\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}; \mathbf{x}_1, \ldots, \mathbf{x}_n) \tag{2.9}$$

where $\mathbf{x}_1, \ldots, \mathbf{x}_n$ is the additional input sequence. The most popular application of encoder-decoder models is machine translation, where to convert some text from French to English, the language model predicts the next word of the English sequence given the entirety of the French sequence and the preceedings words of the English sequence.

Most state-of-the-art neural language models uses a variant of the Transformer architecture [167] as the neural network f_θ. All of the research presented in this dissertation uses the Transformer architecture, except for Chapter 3.3, which uses an older LSTM-based recurrent architecture [68].

2.4 ENCODING TEXT INTO A VOCABULARY

For simplicity, the previous sections refer to the input to a language model as a sequence of words, but in practice, neural language models use a variety of different techniques to construct vocabularies of varying granularities. There is no single solution for forming the base units of language (referred to for the remainder of this chapter as "tokens"), and techniques vary significantly across languages. In English, the simplest vocabularies are character-level–each letter of the alphabet and punctuation becomes a token. Historically, word-level vocabularies, where each token corresponds to a word in the dictionary, were most common. Word-level vocabularies can be created by splitting a string on whitespace and punctuation. Since the space of possible words is near-boundless, in practice only the most common tens or hundreds of thousands of words are included in the vocabulary, and all other words are replaced with an out-of-vocabulary token.

In recent years, subword vocabularies have become standard in neural language modeling. Subword vocabularies are formed by choosing a budget (the desired size of the vocabulary), then running an algorithm that joins letters together into larger units, such that the most common character sequences end up as tokens in the vocabulary. While common words such as "cat" or "dog" end up as single tokens in the vocabulary, uncommon words such as hippopotamus end up bring broken into multiple tokens. Several greedy algorithms have been proposed to approximate optimally breaking up a text corpus into subwords, but byte-pair encoding (BPE) is currently the most popular [146]. Typical subword vocabulary sizes are between 32,000 and 50,000 tokens.

Table 2.1 shows the same string under a few different tokenization schemes.

For all of the types of vocabularies discussed, a decision must be made on whether to convert strings to lowercase before vocabulary creation. Removing case allows for a more compact vocabulary, but it also removes potentially useful information about the location of proper nouns.

Subword vocabularies were designed to be a compromise between the advantage and disadvantages of word-level and character-level vocabularies. Character-level vocabularies are usually very small, no more than a couple hundred tokens. However, the vocabulary can cover near every possible string a person could write.

Table 2.1: Examples of the string “A hippopotamus ate my homework." tokenized using three different vocabularies. With the subword tokenizer, the rare word “hippopotamus" gets broken up into multiple tokens. For word-level tokenizers, if the word “hippopotamus” occurred very infrequently in the corpus used to build the vocabulary (or perhaps the writer of the sentenced misspelled it), it would typically get replaced with an out-of-vocabulary token (row 4).

Vocab Type	Example
character-level	`['A', ' ', 'h', 'i', 'p', 'p', 'o', 'p', 'o', 't', 'a', 'm', 'u', 's',' ', 'a', 't', 'e', ' ', 'm', 'y', ' ', 'h', 'o', 'm', 'e', 'w', 'o', 'r', 'k', '.']`
subword-level	`['A', 'hip', '##pop', '##ota', '##mus', 'ate', 'my', 'homework', '.']`
word-level	`['A', 'hippopotamus', 'ate', 'my', 'homework', '.']`
word-level	`['A', '[UNK]', 'ate', 'my', 'homework', '.']`

Word-level vocabularies cannot feasible contain the hundreds-of-thousands of words present in English text. Realistically, only the most common words are kept, and less common ones are replaced with a special `UNK` token. When text is tokenized with character-level vocabularies, the resulting sequences are very long, while word-level tokenization yields shorter sequences since there is just one token per word. Lastly, word-level representations learned by a neural net tend to more meaningful than character-level representations since a word has semantics associated with it that are common across uses. Subword vocabularies adopt the best of both worlds, using word-level tokens for common words but falling back to subword, or in the worst case, character-level, tokenization for uncommon words. This approach eliminates the need for an out-of-vocabulary token and results in tokenized sequence lengths which are somewhere between the two strategies.

2.5 GENERATING TEXT WITH A LANGUAGE MODEL

Neural language models in themselves are not generative. As described in the previous sections, most language models provide a probability distribution for what the next token in the sequence *could* be, given the previous tokens. To perform generation, an algorithm for decoding sequences from these predicted probability distributions is needed. At each step of decoding, the decoding algorithm performs a forward pass on the neural network using the existing prompt text as input, selects a next token based on the neural network's predictions, adds this token to the prompt, and repeats until the desired number of tokens have been generated.

There are many possible strategies for making a token selection at each step. The simplest strategy is to take the arg max of the distribution, greedily picking the token with the highest probability according to the model. This approach is simple but only allows a single generation to be produced for any given prompt. Alternatively, one can randomly sample from the vocabulary, where each vocab item is assigned a sampling probability proportional to its probability predicted by the language model. This method allows for many different sequences to be generated from the same prompt. However, in practice, this tactic results in noisy, low-quality text, as the probability distributions returned by neural language models tend to be very long tailed, and the chance of sampling a word from this long tail is quite high.

Several strategies have been proposed to improve random sampling techniques by reducing the entropy of the distribution before sampling. Introducing a temperature parameter τ into the softmax computation allows us to smoothly shift probability mass from low-scoring item in the vocabulary to high-scoring ones.

$$P(Y_t = i|\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) = \frac{\exp(\mathbf{E}\hat{\mathbf{y}}_t[i]/\tau)}{\sum_j \exp(\mathbf{E}\hat{\mathbf{y}}_t[j]/\tau)} \tag{2.10}$$

Alternatively, one can introduce sparsity intro the distribution by deliberately zeroing out like-likelihood vocabulary items. Top-k random sampling accomplishes this by restricting sampling to only the k most likely tokens at each step. Nucleus sampling, also referred to as top-p random sampling, accomplishes this by restricting sampling at timestep t to the k_t most likely tokens, where k_t is selected such that these tokens cover no more than p% of the probability mass. For all three of these techniques there is a parameter (τ, k, or p) which controls the amount of randomness we want to permit in the generation. Choosing a low value for these parameters results in an increasingly peaky distribution, which, at its extreme, is the same as taking the arg max. Choosing a high value for these parameters results in the distribution that looks closer and closer to the original scores produced by the model. The diversity-quality tradeoff that results from choice of these parameters is described in detail in Chapter 3.4.

Lastly, search methods such as beam search can be used to try and find the most likely overall sequence. Actually computing the most likely sequence is intractable due to the exponential search space, but algorithms such as beam search so a good job of approximating it. In practice, these techniques tend to lead to text that is bland and repetitive, Chapter 3.3 goes into detail about why this is the case.

2.6 CONTROLLABILITY AND TASK-SPECIFIC GENERATION

In the early days of neural language modelling, it was common to train a separate neural language model for each generative task of interest. For example, if one wanted a system capable of producing chatbot dialog, one would train their neural language model on a dialog dataset such as OpenSubtitles [170]. If one wanted a system able to perform text summarization, one would likewise train a model from scratch on a dataset such at Gigaword [121]. At the time, the neural networks being used for these sorts of tasks were relatively small, and training and maintaining one model per task, was mostly feasible.

In 2018, Radford et al. [130] proposed the idea of training a single "General Purpose Transformer (GPT)" model which could be subsequently finetuned for any number of language tasks. The idea of finetuning a more general model for a specific task had already taken off in computer vision, where researchers had shown a convolutional neural network trained on the ImageNet task of classifying the contents of images could be finetuned for tasks ranging from image segmentation to **{TODO: }**.

General purpose language models intended for generative tasks tend to be trained on massive datasets scraped from the internet, such as C4 [132] or The Pile [56]. It is common to use both decoder-only models trained only to predict the next token given the previous ones [131], as well as encoder-decoder architectures trained with a de-noising loss, where the input is a corrupted version of the text, and the task is to recover the uncorrupted text [132, 95]. Finetuning such models has yielded immense success in tasks across the field of natural language processing. Chapter 5.3 focuses on the feasibility of finetuning for the fill-in-the-blank task.

There are however several limitations to the paradigm of pre-training followed by finetuning. As state-of-the-art neural language models increase in number of parameters, the computational expense of finetuning is becoming increasingly prohibitive. Furthermore, the need to store (potentially in GPU memory) one set of model weights per task makes it very difficult to build downstream applications which need to perform several different tasks. In addition, finetuning only works where there is enough data to fine-tune one. Overfitting is a significant challenge in low-resource settings, where there may only be a handful of training examples.

For these reasons, various approach have been proposed for replacing the finetuning step with methods which require either no or minimal weight training. Brown et al. [23] introduce the technique of few-shot prompting. By constructing a textual prompt which contains several examplars of the goal task, a general-purpose language model can be made to perform the task. Chapter **??** uses this technique for the task of textual style, while Chapter **??** shows how it can be used for a variety of story editing operations. Lester et al. [94] introduce *prompt tuning* as an improvement over few-shot prompting that trains a small neural network to produce an optimal prompt in embedding-space for the goal task.

3 FACTORS THAT IMPACT GENERATED TEXT QUALITY, DIVERSITY, AND DETECTABILITY

3.1 MOTIVATION

It is not a question that natural language generation systems have made leaps and bounds in the last decade. But how do we evaluate this progress and determine whether one NLG system on average produces better outputs than another NLG system? In my research, I show how "better" is difficult to define, and I propose the task of detection–identifying whether a piece of text was written by a human or generated by a system–as a way to better understand and compare the strengths of different NLG systems.

It turns out that choosing the choice of decoding strategy (see Section **{TODO: }**) has a huge impact on the quality of the text that gets generated with a neural language model. Search-based decoding strategies which optimize for choosing the mostly likely overall sequence end up producing text which is much less diverse than text a human writer would produce. This may be tolerable for applications where diversity isn't strictly needed, such as machine translation, but it is problematic for applications where there are many valid ways to continue a prompt. In this chapter, I present investigations of how the choice of decoding strategy creates a trade-off between the human-assessed quality of generated text and the amount of lexical diversity present in said text.

In Section 3.3, we consider search-based decoding strategies, focusing especially on diversity-promoting modifications of beam search. We compared several methods which had previously never been compared with each other, and we showed that none of these diversity-promoting methods improve diversity without serious cost to generation quality. No method outperformed the others on both diversity and quality.

In Section 3.4, we focus on probabalistic generation methods which randomly sample a next word using the distribution predicted by the language model. The simplest strategy is to sample directly from the predicted distribution. However, when many low-likelihood words cumulatively contain quite a bit of probability mass, choosing one of these words can lead to odd or contradictory phrases and semantic errors. Humans readers are quick to notice these types of errors. Thus, more commonly, methods are used that reduce the entropy of the distribution before sampling, which improves generation quality at the cost of diversity. We show that humans have a hard type identifying that text is machine-generated when sampling is heavily restricted to only high-likelihood words.

Finally in Section 3.5, we conduct a large-scale study of the detectability of generated text by human annoators, expanding upon the small-scale human evaluation experiments described in Section 3.4.

3.2 TERMINOLOGY

3.2.1 Diversity

The term "diversity" has been used in the language model literature to refer to a diverse set of properties. Some use it as a synonym for sentence interestingness or unlikeliness [65]. Others consider diversity a measure of how different two or more sentences are from each other [169, 60]. In some framings, diversity is measured across a set of generations coming from the same prompt. Given a particular prompt or input, the goal is to measure the breadth of possible generations the model will produce [113]. Diversity can also be measured as a corpus-level: given all the sentences generated by the model for all prompts, what is the overall lexical diversity **{TODO: citation}**?

In Section 3.3, we define diversity as the ability of a generative method to create a set of possible outputs that are each valid given a particular input but vary as widely as possible in terms of word choice, topic, and meaning. We also use human evaluation to measure generation interestingness. In Section 3.4, we consider decoder-only language models, where no additional input is conditioned on. In this setting, we instead consider corpus-level diversity across all the model's generations. In both sections, diversity of a set of model generations is automatically measured using distinct-n, the number of unique n-grams within the set divided by the total number of n-grams.

3.2.2 Quality

"Quality" is also a difficult property to define. When measured in downstream applications, it can be quantified as how many times a user interactions with the generative system (for example, the number of conversation turns with a dialog system) before losing interest. It can also be evaluated directly by asking a human rater, "how good is this text?", though definitions of "good" vary widely across the literature **{TODO: add citation}**.

To some extent, quality can also be measured automatically. In tasks with a clear goal, like machine translation or summarization, one can compare the generation against a gold standard. Generally, quality is strongly associated with fluency, and in **{TODO: citation}** we show that up until a point, the lower perplexity text is assigned by a language model, the more fluent it is.

Sections 3.3 and 3.4 take quite different approaches to measuring generation quality. In 3.3, we consider a dialog model, and ask humans to assess the quality of model responses on three axes: fluency, adequacy, and interestingness. In 3.4, we propose a novel method for assessing generation quality based on the premise that humans (or a trained discriminator) ought to have a hard time distinguishing between real human-written text and model outputs when the model outputs text that is high-quality.

3.3 DIVERSITY-PROMOTING SEARCH-BASED DECODING METHODS

3.3.1 Introduction

In 2018, state-of-the-art neural language models were based on recurrent neural networks, such as LSTMs, that struggled to generate sequences that were both long and high-quality. However, these models were beginning to have huge success over statistical techniques in natural language processing tasks such as machine translation [156, 110], text summarization [120], and dialog systems [170].

At the time, the standard convention for generation was to try to generate the most likely overall sequence from the language model. This approach made a lot of sense in machine translation, where generating one correct translation was more important than generating diverse translation. Since computing the overall most likely output sequence is intractable, early work in neural machine translation found that beam search was an effective strategy to heuristically sample sufficiently likely sequences from these probabilistic models [156]. However, as neural language models came to be applied increasingly to open-ended tasks, beam search was found to be ill-suited to generating a set of diverse candidate sequences; this is because candidates outputted from a large-scale beam search often only differ by punctuation and minor morphological variations [97].

There are a number of reasons why it is desirable to produce a set of diverse candidate outputs for a given input. For example, in collaborative story generation, the system makes suggestions to a user for what they should write next [34]. In these settings, it would be beneficial to show the user multiple different ways to continue their story. In image captioning, any one sentence-long caption is probably missing some information about the image. Krause et al. [86] show how a set of diverse sentence-length image captions can be transformed into an entire paragraph about the image. Lastly, in applications that involve reranking candidate sequences, the reranking algorithms are more effective when the input sequences are diverse. Reranking diverse candidates has been shown to improve results in both open dialog and machine translation [96, 97, 60]. Furthermore, in open-ended dialog, the use of reranking to personalize a model's responses for each user is a promising research direction [32].

With these sorts of applications in mind, a variety of alternatives and extensions to beam search were proposed which sought to produce a set of diverse candidate responses instead of a single high likelihood one

[96, 169, 90, 157]. Many of these approaches show marked improvement in diversity over standard beam search across a variety of generative tasks. However, as of 2018, there had been little attempt to compare and evaluate these strategies against each other on any single task.

In this sub-chapter, we survey methods for promoting diversity during decoding in order to systematically investigate the relationship between diversity and perceived quality of output sequences. We focus on conditional language models–models that are trained to map from some input, perhaps an image or some text, to a target sequence. In addition to standard beam search and greedy random sampling, we compare several recently proposed modifications to both methods. We present a detailed comparison of existing diverse decoding strategies on two tasks: open-ended dialog and image captioning, and recommendations for a diverse decoding strategy.

3.3.2 Diverse Decoding Strategies

There are many ways to decode text from a conditional language model. Let $\mathbf{y}$ represent the sequence of tokens for the target sequence ($\mathbf{y} = y_1 \ldots y_n$) and $\mathbf{x}$ represent the input (which may be another token sequence, or, in the case of image captioning, an image). Most decoding strategies strive to find the most likely overall sequence, i.e. pick a $\hat{\mathbf{y}}$ such that:[1]

$$\hat{\mathbf{y}} = \arg\max_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}) = \arg\max_{\mathbf{y}} \prod_{t=1}^{N} P(y_t \mid y_{<t}, \mathbf{x})$$

In practice, this can't be computed exactly, no sub-exponential algorithm exists for find the optimal decoded sequence, and thus the field instead use approximations.

Beam search approximates finding the most likely sequence by performing breadth-first search over a restricted search space. At every step of decoding, the method keeps track of b partial hypotheses. The next

1 The formulation we are using here is slightly different than Section **{TODO: }**, in that we use $\mathbf{y}$ and $\hat{\mathbf{y}}$ to refer to sequence of tokens, rather than framing in terms of a sequence of embedding vectors.

set of partial hypotheses are chosen by expanding every path from the existing set of b hypotheses, and then choosing the b with the highest scores. If the goal is to approximate finding the most likely over-all sequence, the log-likelihood of the partial sequence is used as the scoring function. Algorithm 1 gives an overview of the beam search algorithm.

Algorithm 1 Beam Search Inference

```
procedure BEAM SEARCH
    B ← {SOS}
    k ← BeamWidth
    out ← k-best output list
    while |out| < k do
        front ← remove all nodes from B
        for w ∈ front do
            succ ← w's k-best successors
            for s ∈ succ do
                if s == EOS then
                    out ← out ∪ {s}
                else
                    B ← B ∪ {s}
                end if
            end for
        end for
        Sort B
        if |B| > k then
            Prune B to k-best successors
        end if
    end while
     return out
end procedure
```

Since beam search only explores a limited portion of the overall search space, it tends to yield multiple variants of the same high-likelihood sequence, sequences that often only differ in punctuation and minor morphological changes [97]. Therefore, standard beam search is not ideal for producing diverse outputs. In this section, we will discuss a variety of methods that have been developed recently to eliminate redundancy during decoding and generate a wider range of candidate outputs.

NOISY PARALLEL APPROXIMATE DECODING Introduced by Cho [31], NPAD is a technique than can be applied to any decoding setting. The main idea is that diversity can be achieved more naturally by taking advantage of the continuous manifold on which neural nets embed language. Instead of encouraging diversity by manipulating the probabilities outputted from the model, diverse outputs are instead produced by adding small amounts of noise to the hidden state of the decoder at each step. The noise is randomly sampled from a normal distribution. The variance is gradually annealed from a starting σ_0 to 0 as decoding progresses (that is $\sigma_t = \frac{\sigma_0}{t}$) under the reasoning that uncertainty is greatest at the beginning of decoding. NPAD can be used in conjunction with any decoding strategy; following the best results from the original paper, we show results using NPAD with beam search.

Extensions to NPAD have sought to learn the direction in which to manipulate the hidden states using an arbitrary decoding objective [62]. Since such objectives can be highly domain-specific, we do not evaluate this method.

TOP-g CAPPING In beam search, it is often the case that one hypothesis h is assigned a much higher probability than all other hypotheses, causing all hypotheses in the next step to have h as their parent. Following Li and Jurafsky [97] and Li et al. [98], we add an additional constraint to standard beam search to encourage the model to choose options from diverse candidates. At each step t, current hypotheses are grouped according to the parental hypothesis they come from. After grouping candidates, only the top g from each grouping are considered. The resulting $b \times g$ candidates are ranked, and the top b are selected as hypotheses for the next beam step.

HAMMING DIVERSITY REWARD Vijayakumar et al. [169] proposes adding an additional diversity-promoting term, θ, to the log-likelihood before reranking. This term measures how different a candidate hypothesis $c_{\leq t}^{(i)}$ is from the partial hypotheses selected in the previous step. Let $\mathcal{H}_{t-1} = \{c_{\leq t-1}^{(1)}, \ldots c_{\leq t-1}^{(b)}\}$ be these partial hypotheses. Then the beam search scoring function for the ith candidate at timestep t becomes:

$$\begin{aligned}\text{score}(c_{\leq t}^{(i)}) = \sum_{j=1}^{t} \big(\log P(c_j^{(i)} | c_{<j}^{(i)}, \mathbf{x})\big) \\ + \lambda\theta(c_{\leq t}^{(i)}, \mathcal{H}_{t-1})\end{aligned}$$

where λ is a tunable hyperparameter. Vijayakumar et al. [169] try a variety of definitions for θ, including embedding diversity and n-gram diversity, but they find that Hamming distance, the number of tokens in the candidate sequence which exist in the previously selected partial hypotheses, is most effective. We take the negative of the Hamming distance as θ.

ITERATIVE BEAM SEARCH In an attempt to improve the size of the search space explored without sacrificing runtime, Kulikov et al. [90] propose an iterative beam search method. Beam search is run many times, where the states explored by subsequent beam searches are restricted based on the intermediate states explored by previous iterations. Formally, we can define the set of all partial hypotheses for beam search instance i at time step t as $\mathcal{H}_t^{(i)}$. From here, the search space explored by beam search instance i can be expressed as $S_i = \cup_{t=1}^{T} \mathcal{H}_t^{(i)}$. The ith beam search is prevented from generating any partial hypothesis that has previously been generated, that is, any hypothesis found in $S_{<i} = \cup_{i'=0}^{i-1} S_{i'}$.

The authors also attempt a soft inclusion criterion, where any states within ϵ Hamming distance from a previously explored state are also excluded. During the experimentation of Kulikov et al. [90], however, the soft-inclusion was found to not be beneficial; thus, we only restrict exact matches of previous states in our implementation. In practice, this means after the first beam search instance runs as normal, the first step of the second beam search instance will contain the b+1 to 2b-most likely starting tokens; this pattern holds for the third beam search instance, and so on.

CLUSTERED BEAM SEARCH Most recently, Tam et al. [157] proposed a clustering-based beam search method to help condense and remove meaningless responses from chatbots. Specifically, at each decoding step t, this method initially considers the top $2 * b$ candidates. From there, each candidate sequence is embedded[2], and the embeddings are clustered into c clusters using K-means. Finally, we take the top $\frac{b}{c}$ candidates from each cluster. Note that in the case any clusters have size less than $\frac{b}{c}$, we then include the highest-ranked candidates not found after clustering.

CLUSTERING POST-DECODING (PDC) All the previous methods modified the decoding algorithm to encourage diversity. However, it is also possible to encourage additional diversity post-hoc by sampling several generations and then choosing the most diverse one. On the task of sentence simplification, after decoding using a large-scale diversity-promoting beam search (beam size 100), Kriz et al. [88] then clustered similar sentences together to further increase the variety of simplifications from which to choose. Document embeddings generated via Paragraph Vector [91] were used as the sentence embeddings with which to perform K-means.

In this work, we extend this post-decoding clustering idea in three key ways. First, we make use of sentence-level embeddings which leverage the pre-trained language representations from the Bidirectional Encoder Representations from Transformers (BERT) [42].[3] Second, after clustering, Kriz et al. [88] took the sentence closest to the centroid of each cluster as the representative candidate; we instead choose the highest-ranked candidate (according to log-likelihood) from each cluster to ensure the best candidates are still selected. Finally, after performing standard K-means clustering, we found that it was often the case that some clusters contained large numbers of good candidates, while others contained very few candidates that are also either ungrammatical or otherwise inferior. Thus, in our implementation, we remove clusters containing two or fewer sentences, and then sample a second candidate from each of the remaining clusters, prioritizing selecting candidates from larger clusters first.

---

2 We follow Tam et al. [157] and used averaged GloVe word embeddings [127].
3 BERT sentence-level embeddings were obtained using https://github.com/hanxiao/bert-as-service.

3.3.3 Experimental Setup

We evaluate the decoding strategies described in the previous section under the following settings. For each of the published beam search algorithms, we choose the hyperparameters that were found to be best in the original publications.

Method	Setting
RS	Random sampling with temp = 0.5, 0.7, 1.0, or 1.0 with top-10 capping.
Standard BS	Standard beam search
Top5Cap BS	Top-g capping with $g = 3$
Iter5 BS	Iterative beam search with 5 iterations
HamDiv0.8 BS	Hamming Diversity with $\lambda = 0.8$
Cluster5 BS	Clustered beam search with 5 clusters
NPAD0.3 BS	Noisy Decoding with $\sigma_0 = 0.3$

For random sampling, we sample 10 outputs, and with beam-search based methods, we use a beam size of 10 to generate 10 outputs. In addition, we show results from oversampling then filtering. We use a beam size of 100 or generate 100 samples through random sampling, and then we select 10 from the 100, either through post-decoding clustering (PDC) or by taking the 10 candidates with highest likelihood.

We examine these decoding strategies on two tasks: open ended dialog and image captioning. For each task, we evaluate both the quality and diversity of the 10 outputs from each strategy.

Open-ended Dialog Task

In the dialog domain, we use an LSTM-based sequence-to-sequence (Seq2Seq) model implemented in the OpenNMT framework [84]. We match the model architecture and training data of Baheti et al. [11]. The Seq2Seq model has four layers each in the encoder and decoder, with hidden size 1000, and was trained on a cleaned version of OpenSubtitles [162] to predict the next utterance given the previous one.

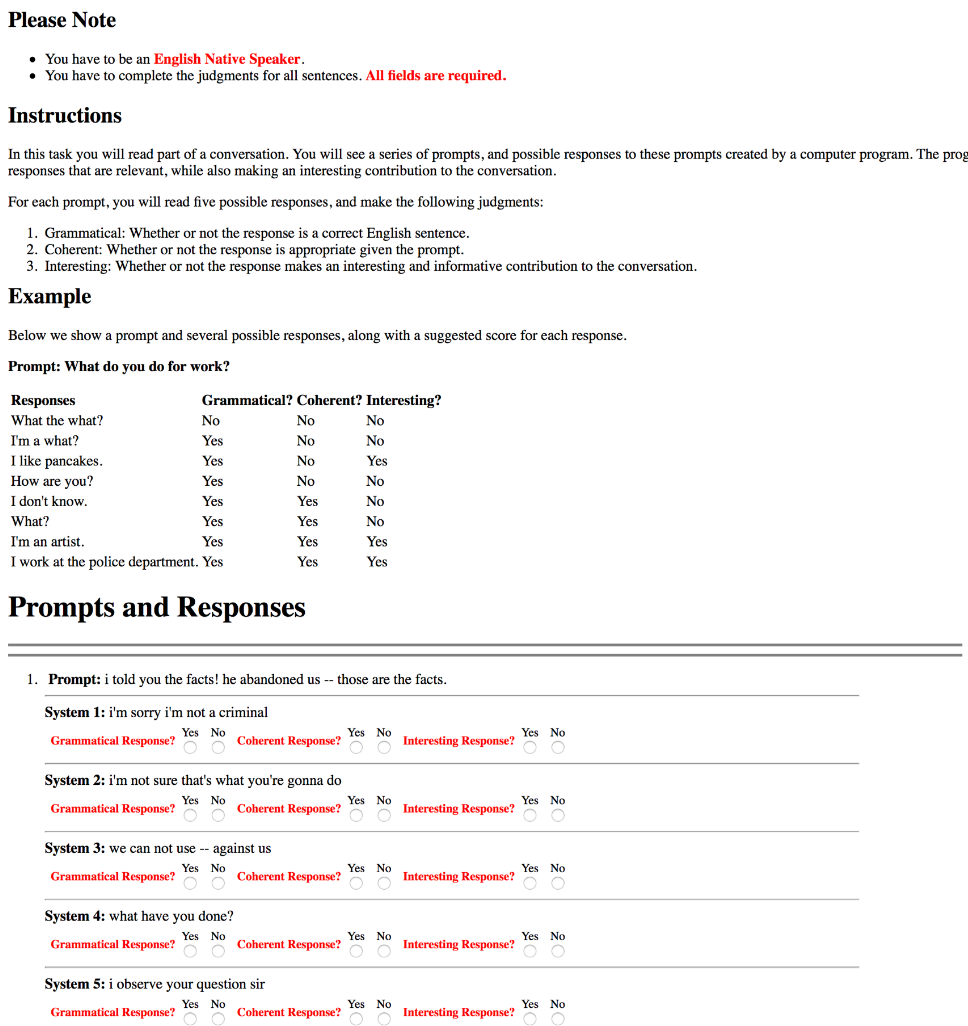
Method	Strategy	Fluency	Adequacy	Interestingness	Ppl	Dist-1	Dist-2	Ent-2	Ent-4
Reference	–	0.795	0.732	0.636	–	–	–	–	–
RS 0.5	10	0.780	0.446	0.383†	52.46	0.54	0.70	3.84	3.70
	100	**0.928**	0.541†	0.329	19.87	0.50	0.63	3.46	3.20
	100,PDC	0.844	0.521	0.372	29.53	0.57	0.70	3.65	3.34
RS 0.7	10	0.758	0.399	0.388†	35.98	0.63	0.80	4.08	3.84
	100	0.911	0.547†	0.329	9.72	0.53	0.68	3.56	3.20
	100,PDC	0.850	0.488	0.366	14.64	0.64	0.76	3.75	3.31
RS 1.0	10	0.550	0.303	0.386†	67.99	**0.74**	**0.87**	**4.35**	**4.08**
	100	0.886	0.506	**0.390**	8.66	0.63	0.76	3.73	3.27
	100,PDC	0.797	0.437	0.388†	16.15	0.73†	0.81	3.86	3.36
RS 1.0,top10	10	0.745	0.418	0.387†	10.33	0.60	0.80	4.12	3.91
	100	0.922	**0.548**	0.347	**5.10**	0.52	0.68	3.54	3.18
	100,PDC	0.852	0.494	0.372	6.96	0.63	0.76	3.74	3.27

Table 3.1: Results for all random sampling methods that were evaluates. These include temperatures 0.5, 0.7, and 1.0, as well as Top-10 random sampling. For each method, we show results with for generating 10 samples; generating 100 samples followed by selecting the 10 most likely outputs; and generating 100 samples followed by post-decoding clustering to select 10 outputs. In each section, the highest value is bolded, and statistical ties are marked †.

Evaluation is performed on 100 prompts from the Cornell Movie Dialog Corpus [40]. These prompts are a subset of the 1000 prompts used in Baheti et al. [11], which were filtered using item response theory for discriminative power.

We report perplexity (PpL), averaged over *all* the top 10 outputs for each example.[4] Since the quality of open-ended dialog is notoriously difficult to evaluate automatically, we ran a human evaluation task on Amazon Mechanical Turk where annotators were shown a prompt and 5 potential responses generated by any of our decoding methods. Evaluators were asked to provide binary ratings on fluency, adequacy, and interestingness for each response. Overall, we collected 3 human judgments for each of the top ten responses for each of our decoding methods; in other words, we collected 3,000 judgments per method. Figure 3.1 shows the instructions given to the human raters.

4 This differs from existing work which computes perplexity over only the top output for each example. For our task we are interested in the quality of all of the generated responses.



Please Note

- You have to be an **English Native Speaker**.
- You have to complete the judgments for all sentences. **All fields are required.**

Instructions

In this task you will read part of a conversation. You will see a series of prompts, and possible responses to these prompts created by a computer program. The program attempts to generate responses that are relevant, while also making an interesting contribution to the conversation.

For each prompt, you will read five possible responses, and make the following judgments:

1. Grammatical: Whether or not the response is a correct English sentence.
2. Coherent: Whether or not the response is appropriate given the prompt.
3. Interesting: Whether or not the response makes an interesting and informative contribution to the conversation.

Example

Below we show a prompt and several possible responses, along with a suggested score for each response.

Prompt: What do you do for work?

Responses	Grammatical?	Coherent?	Interesting?
What the what?	No	No	No
I'm a what?	Yes	No	No
I like pancakes.	Yes	No	Yes
How are you?	Yes	No	No
I don't know.	Yes	Yes	No
What?	Yes	Yes	No
I'm an artist.	Yes	Yes	Yes
I work at the police department.	Yes	Yes	Yes

Prompts and Responses

1. **Prompt:** i told you the facts! he abandoned us -- those are the facts.

System 1: i'm sorry i'm not a criminal
Grammatical Response? Yes No **Coherent Response?** Yes No **Interesting Response?** Yes No

System 2: i'm not sure that's what you're gonna do
Grammatical Response? Yes No **Coherent Response?** Yes No **Interesting Response?** Yes No

System 3: we can not use -- against us
Grammatical Response? Yes No **Coherent Response?** Yes No **Interesting Response?** Yes No

System 4: what have you done?
Grammatical Response? Yes No **Coherent Response?** Yes No **Interesting Response?** Yes No

System 5: i observe your question sir
Grammatical Response? Yes No **Coherent Response?** Yes No **Interesting Response?** Yes No

Figure 3.1: The full instructions for our Amazon Mechanical Turk task to evaluate the quality of our dialog system responses.

Method		Fluency	Adequacy	Interestingness	Ppl	Dist-1	Dist-2	Ent-2	Ent-4
Reference		0.795	0.732	0.636	–	–	–	–	–
RS 0.7	(sample 10)	**0.758**	0.399	**0.388**	35.98	0.63	0.80	4.08	3.84
RS 1.0	(sample10)	0.550	0.303	0.386†	67.99	**0.74**	**0.87**	**4.35**	**4.08**
RS 1.0,top10	(sample 10)	0.745†	**0.418**	0.387†	**10.33**	0.60	0.80	4.12	3.91
Standard BS	(10 beams)	**0.950**	**0.621**	0.336	**4.01**	0.37	0.45	3.16	3.01
Top3Cap BS	(10 beams)	0.942†	0.603	0.346	4.03	0.37	0.46	3.17	3.03
Iter5 BS	(10 beams)	0.903	0.520	0.335	5.42	**0.62**	**0.74**	**3.68**	**3.25**
HamDiv0.8 BS	(10 beams)	0.923	0.599	0.366†	4.56	0.33	0.37	3.08	3.00
Cluster5 BS	(10 beams)	0.936	0.582	**0.381**	4.23	0.39	0.46	3.24	3.06
NPAD0.3 BS	(10 beams)	0.942†	0.604†	0.335	4.05	0.36	0.44	3.13	2.99
RS 1.0,top10	(sample 100, rank)	**0.922**	**0.548**	0.347	**5.10**	0.52	0.68	3.54	3.18
RS 1.0,top10	(sample 100, PDC)	0.852	0.494	**0.372**	6.96	**0.63**	**0.76**	**3.74**	**3.27**
Standard BS	(100 beams, rank)	**0.964**	**0.611**	0.332†	**4.01**	0.44	0.61	3.33	3.05
Standard BS	(100 beams, PDC)	0.944	0.599	**0.346**	4.42	**0.57**	**0.70**	**3.59**	**3.21**

Table 3.2: Results on 100 dialog prompts. The first row shows the mean human ratings of the single reference response available for each prompt. The next three rows show results for random sampling, with 10 samples drawn per prompt. The next six rows are variants of beam search using beam size 10. The last four rows use random sampling or standard beam search to generate 100 outputs, then filter down to 10 outputs either through ranking by log-likelihood or by performing post-decoding clustering (PDC). In each section, the highest value is bolded, and statistical ties are marked †.

Image Captioning Task

For image captioning, we use a state-of-the-art model introduced in Anderson et al. [7]. We take advantage of Luo [109]'s open-source implementation and released model parameters trained on MSCOCO [104]. We evaluate on a test set containing 5000 images.

We report Semantic Propositional Image Caption Evaluation (SPICE) scores, an automatic evaluation metric that has been shown to correlate well with human judgments of quality[6]. SPICE measures how well the semantic scene graph induced by the proposed caption matches one induced by the ground truth. In addition to computing SPICE on the top-scoring caption (SPICE@1), we follow Vijayakumar et al. [169] in reporting Oracle SPICE@10 scores. This is done to show the upper bound on the potential impact diversity can have. We also compute the mean SPICE score across all of the candidate captions for an image. Unlike SPICE@1 and SPICE@10, this metric shows the overall quality of *all* of the candidate captions, which is useful to know for applications that combine diverse candidate output sequences [86].

Method		SPICE Mean	SPICE @1	SPICE @10	Dist-1	Dist-2	Ent-2	Ent-4
RS 0.7	(sample10)	**0.170**	**0.192**	**0.278**	0.31	0.52	3.67	4.00
RS 1.0	(sample10)	0.133	0.167	0.247	**0.44**	**0.71**	**4.17**	**4.26**
RS 1.0,top10	(sample10)	0.159	0.183	0.272	0.33	0.59	3.90	4.17
Standard BS	(10 beams)	0.194	0.193	0.283	0.18	0.26	2.94	3.18
Top3Cap BS	(10 beams)	**0.195**	**0.196**	0.282	0.17	0.26	2.93	3.17
HamDiv0.8 BS	(10 beams)	0.194	0.194	0.282	0.18	0.27	2.98	3.19
Cluster5 BS	(10 beams)	0.191	0.194	**0.285**	**0.19**	**0.28**	**3.04**	**3.25**
NPAD0.3 BS	(10 beams)	0.191	0.192	0.280	0.18	0.26	2.94	3.17
RS 1.0,top10	(sample100, rank)	**0.182**	**0.188**	**0.284**	0.25	0.41	3.31	3.64
RS 1.0,top10	(sample100, PDC)	0.169	**0.188**	0.282	**0.31**	**0.52**	**3.62**	**3.91**
Standard BS	(100 beams, rank)	**0.188**	0.190	0.279	0.20	0.31	3.04	3.32
Standard BS	(100 beams, PDC)	0.186	**0.192**	**0.288**	**0.24**	**0.38**	**3.25**	**3.57**

Table 3.3: Image captioning results for selected random sampling and beam search methods. SPICE@1 measures the SPICE score of the most likely caption. SPICE@10 is the maximum score across the 10 candidates generated by each method. Mean SPICE is the mean score over all 10 candidates. In each section, the best value is bolded.

Evaluating Diversity

To measure the diversity across the generated candidate sequences for a given input, we report **Dist-k**, the total number of distinct k-grams divided by the total number of produced tokens in all of the candidate responses for a prompt [96]. We report Dist-2 and Dist-4 averaged over the prompts in the test set.

A limitation of Dist-k is that all k-grams that appear at least once are weighted the same, ignoring the fact that infrequent k-grams contribute more to diversity than frequent ones. Zhang et al. [192] instead propose an entropy metric, **Ent-k**, defined as:

$$\textit{Ent-k} = \frac{-1}{\sum_{w \in S} F(w)} \sum_{w \in S} F(w) \log \frac{F(w)}{\sum_{w' \in S} F(w')}$$

where S is the set of all k-grams that appear in candidate responses for an example, and $F(w)$ denotes the frequency of w in the candidate responses.

3.3.4 Results

We report results on dialog systems and image captioning in Tables 3.2 and 3.3, respectively. As expected, random sampling-based approaches yield outputs with greater diversity but worse quality than beam search-based approaches. Over-sampling then filtering increases the quality of outputs while still ensuring high diversity. In the following sections, we discuss the diversity-quality tradeoff, and then delve further into the results for each method group.

The Quality Diversity Tradeoff

The goal of diverse decoding strategies is to generate high-quality candidate sequences which span as much of the space of valid outputs as possible. However, we find there to be a marked trade-off between diversity and quality. This can be seen in Figure 3.2, where we plot the human-judged quality score for each dialog

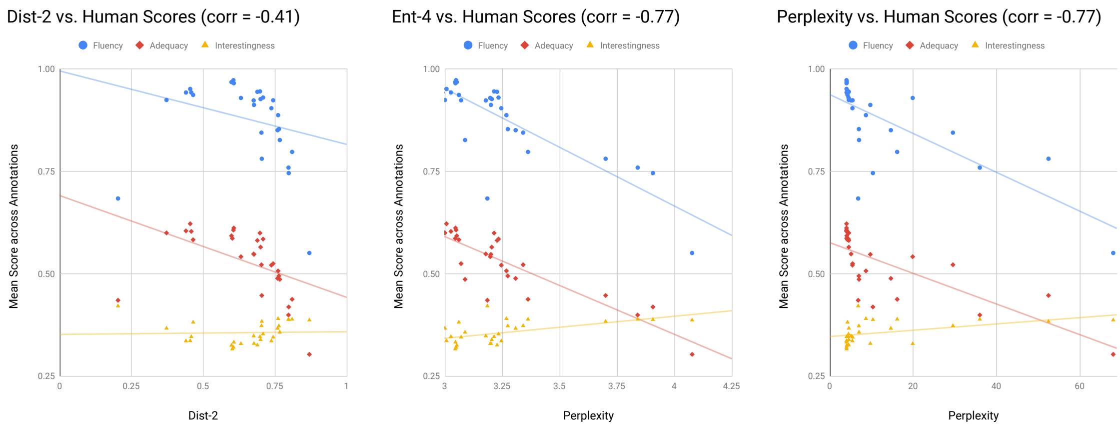


Figure 3.2: Each decoding strategy is plotted, showing that human-perceived quality is negatively correlated with diversity. The Pearson Correlation coefficients between each statistic and the average of fluency, coherence, and interestingness are shown in parentheses.

experiment against our primary diversity descriptive statistics. Fluency and adequacy are both strongly negatively correlated with diversity. While we had expected interestingness to be positively correlated with diversity, the fact that it is not suggests that existing diversity statistics are insufficient for capturing what it means to humans for outcomes to be interesting.

Likewise, in image captioning, the mean SPICE score of the 10 candidate captions (averaged over all examples for each experimental setting) is strongly anti-correlated with diversity, with a Pearson correlation coefficient of -0.83 with the Ent-4 measure and -0.84 with Dist-2. Clearly it remains an open challenge to generate a diverse set of image captions that are all high-quality.

When researchers choose to use a diverse decoding strategy, they must decide where on the quality-diversity tradeoff they would like to lie; selecting an optimal method depends strongly on one's tolerance for errors. In machine translation, where mistakes could severely impact coherence, beam search-based methods, which tend to result in better fluency and coherence, but worse diversity might be preferred. In more open-ended applications, where novel text is of greater importance, increased diversity could be worth the fluency and coherency hit. As state-of-the-art models continue to improve, one would hope that the quality cost of encouraging diversity will continue to decrease.

In the interest of reporting a single overall best method for each task, we computed a sum-of-ranks score for each method. For dialog, we ranked the methods each by fluency, coherence, interestingness, and Ent-4, and then took a weighted sum of the four ranks, with 50% of the weight assigned to Ent-4, and 50% distributed evenly among the human evaluation ranks. Overall, clustered beam search and standard BS (beam size 100, PDC) have the best scores, followed by clustered beam search (beam size 10). Similarly, for image captioning, we rank the methods by their mean SPICE score and by Ent-4. Summing these ranks, random sampling (temp 1.0, top-10 capping, PDC) came in first. Standard beam search, Hamming Diversity beam search, and Top-g capping beam search (beam size 10) tied for second.

3.3.5 Random Sampling-based Methods

Higher sampling temperatures result in both an increase in diversity in generated responses and a reduction in overall quality. In the dialog domain, evaluators consistently rate the responses sampled with temperature 1.0 to have worse fluency, coherence, and interestingness when those sampled with temperature 0.5. In the image captioning domain, lower temperature improves automatic evaluation metrics for quality while reducing diversity.

For dialog, restricting sampling to the top-10 vocabulary words is a more effective strategy than adjusting temperature for ensuring balance between the quality and diversity of outputs. Top-10 random sampling has the highest fluency, coherence, and interestingness, as well as significantly lower perplexity than other random sampling methods. However, this trend did not extend to image captioning, where top-10 random sampling results in both worse SPICE scores and lower diversity measures than setting the temperature to 0.7. This may be because image captioning is a less ambiguous task than open-ended dialog, leading to a better-trained model that puts more probability mass on high-quality vocabulary words, ameliorating the challenge top-c filtering is designed to eliminate: that of a long tail of low probability vocabulary words taking up a large amount of probability mass.

Beam Search-based Methods

For dialog, clustered beam search (Cluster5 BS) performs the best of all beam search methods in terms of human-judged interestingness. It ties for best with NPAD0.3BS on fluency and ties with Standard BS on coherence. Iterative beam search (Iter5 BS) achieves the greatest diversity, but at the expensive of quality. It has the lowest human-judged coherence among beam search methods; thus, we do not evaluate this method on image captioning. For image captioning, Cluster5 BS has the highest diversity among beam search methods, but this difference is quite small. Cluster5 BS also has the highest SPICE@10 score, indicating it is the best method for generating at least one high quality candidate. However, Top3Cap BS results in the highest mean SPICE score, suggesting it is best at ensuring all outputs are reasonable quality.

Effect of Over-sampling

In our experiments, we explore over-sampling 100 outputs, and then either using post-decoding clustering (PDC) or re-ranking by log-likelihood to filter these 100 down to 10 diverse outputs.

In the dialog domain, this over-sampling approach is a definite win. When over-sampling with random sampling both methods of filtering substantially improve human judgements of fluency and adequacy compared to random sampling only 10 outputs. However, interestingness scores go down, and while the outputs are still more diverse than beam search-based methods, they are less diverse than random sampling without filtering. In the beam search methods that use a beam size of 100 then filter down to 10, human-judged quality is on par with beam size 10 results, but diversity is considerably higher.

When comparing the two types of filtering, PDC results in higher interestingness and diversity statistics, while log-likelihood re-ranking improves fluency and adequacy. This again demonstrates the trade-off between quality and diversity.

Table 3.1 shows the results with every method where we generate 10 samples; generate 100 samples followed by selecting the 10 most likely outputs; and generate 100 samples followed by post-decoding clustering to select 10 outputs. We see that **{TODO: }**

For image captioning, over-sampling with reranking does not consistently improve quality as it does in the dialog domain. Mean SPICE score is improved for random sampling but not for beam search. SPICE@1 becomes worse for both random sampling and decoding, while SPICE@10 improves for random sampling, and for beam search when PDC is applied. From these results, we can conclude that over-sampling then ranking does not have a sizeable effect, either negative or positive, on quality. Moreover, the diversity of the captions generated by random sampling actually decreases when oversampling. The diversity of beam search-generated captions does improve with over-sampling.

While oversampling does generally improve outcomes on the diversity/quality tradeoff, it is more computationally expensive, particularly with beam search. Running PDC also requires generating sentence embeddings for every output, which adds additional computation time.

3.3.6 Summary of Contributions

The work described in this section was published in the 2019 Proceedings of the Association of Computational Linguistics [74]. The work was performed with in conjunction with Reno Kriz, with the assistance of Maria Kustikova and the mentorship of João Sedoc and Chris Callison-Burch. Reno and I contributed equally to the development of the ideas in the paper and the design and implementation of the experiments.

3.4 IMPACT OF DECODING STRATEGY ON THE DETECTION OF MACHINE-GENERATED TEXT

State-of-the-art generative language models are now capable of producing multi-paragraph excerpts that at a surface level are virtually indistinguishable from human-written content [187, 131, 2]. Often, only subtle logical fallacies or idiosyncrasies of language give away the text as machine-generated, errors that require a close reading and/or domain knowledge for humans to detect.

Deceptive text, whether human- or machine-generated, has entered the sphere of public concern [37]. It propogates quickly [171], sets political agendas [165], influences elections [5], and undermines user trust [174, 152]. Recently, Adelani et al. [2] have shown that automatically generated reviews are perceived to be as fluent as human-written ones. As generative technology matures, authors, well-meaning or otherwise, will increasingly employ it to augment and accelerate their own writing. However, there has thus been little inquiry into the textual properties that cause humans to give generated text high human-like ratings compared to those that cause automatic systems to rate it highly.

This task of trying to guess whether text is coming from a robot or a fellow human was made famous by the Turing Test [164]. It continues to be used is chatbot evaluation [107]. The related (but not identical) task of asking human raters to judge the quality of machine-generated excerpts remains the gold-standard for evaluating open-domain generation systems [92].

In turns out that even with a fixed language model, choice of decoding strategy has a huge impact on the detectability of generated text. Using top-k random sampling, a decoding method where only the selection of high-likelihood words is permetted, means we are less likely to make a poor choice and create the type of mistakes that are easy for humans to detect. Since humans are not proficient at identifying when a model subtly favors some utterances more often than a human author would, they don't notice the over-representation of high-likelihood words in the generated text. In contrast, automatic systems excel at identifying statistical anomalies and struggle to build deeper semantic understanding. Top-k in particular creates text that is easy for machines to detect but very hard for humans. Thus, we observe the general trend: as the number of unlikely words available to be chosen is increased, humans get *better* at detecting fakes while automatic systems get *worse*.

In this work, we study three popular random decoding strategies—top-k, nucleus, and temperature sampling—applied to GPT-2 [131]. We draw a large number of excerpts generated by each strategy and train a family of BERT-based [42] binary classifiers to label text excerpts as human-written or machine-generated.

We find large differences in human rater and classifier accuracy depending on the decoding strategy employed and length of the generated sequences. Regardless of strategy, we find human raters achieve significantly lower accuracy than the automatic discriminators. We also show that when a decoding strategy severely modifies the unigram token distribution, as top-k does, humans have trouble detecting the resultant generated text, but automatic classifiers find it the easiest to discriminate. Worryingly, we further find that classifiers are brittle; they generalize poorly when trained to discriminate samples from one strategy and then evaluated on samples from another.

3.4.1 Detection as a Task

The rise of machine-generated content has led to the development of automated systems to identify it. GROVER was designed to not only generate convincing news excerpts but to also identify them using a fine-tuned version of the generative model itself [187]. GLTR, expecting attackers to use sampling methods that favor high-likelihood tokens, aims to make machine-generated text detectable by computing histograms over per-token log likelihoods [59]. Bakhtin et al. [12] frame human-text detection as a ranking task and evaluate their models' cross-domain and cross-model generalization, finding significant loss in quality when training on one domain and evaluating on another. Schuster et al. [144] argue that the language distributional features implicitly or explicitly employed by these detectors are insufficient; instead, one should look to explicit fact-verification models. Finally, discriminators for whether text is machine-generated are a promising research direction in adversarial training [102, 99] and in automatic evaluation of generative model quality [125, 80, 107].

We frame the detection problem as a binary classification task: given an excerpt of text, label it as either human-written or machine-generated. In particular, we are interested in how variables such as excerpt length and decoding strategy impact performance on this classification task. We thus create several datasets. Each is approximately balanced between positive examples of machine-generated text and negative examples of human-written text. While they all share the same human-written examples, each dataset contains a different set of machine-generated examples sampled using one particular decoding strategy. We also build additional datasets by truncating all of the examples to a particular sequence length,

By training a separate classifier on each dataset, we are able to answer questions about which decoding strategy results in text that is the easiest to automatically disambiguate from human-written text. We are also able to answer questions about how the length of the examples in the training set impacts our ability to automatically classify excerpts of that same length as either human-written or machine-generated.

3.4.2 Method

Dataset Construction

All of our generated text samples are drawn from GPT-2, a state-of-the-art Transformer-based generative language model that was trained on text from popular web pages [131]. While we use the GPT-2 LARGE model with 774M parameters, we found that similar trends to those reported here hold in experiments with smaller language models.

Given an autoregressive language model that defines a probability distribution over the next token given the previous tokens in a sequence, a decoding strategy generates text by deciding how to output a token at each step based on the predicted distributions. Perhaps the most straightforward decoding strategy is to randomly choose a token with probability proportional to its likelihood. A challenge with the random sampling approach is that these probability distributions often contain a long tail of vocabulary items that are individually low-probability but cumulatively comprise a substantial amount of probability mass. Holtzman et al. [69] observe that choosing tokens from this tail often leads to incoherent generations.

Top-k sampling, nucleus sampling, and (in the extreme) beam search have all been proposed to heuristically promote samples with higher per-token likelihoods. Top-k and nucleus sampling both do so by setting the likelihood of tokens in the tail of the distribution to zero. Top-k restricts the distribution to all but the k most likely tokens, where k is a constant [49]. Nucleus sampling, also called top-p, truncates the distribution at each decoding step t to the k_t-most-likely next tokens such that the cumulative likelihood of these tokens is no greater than a constant p [69].

We thus consider three different decoding strategy settings:

- Sample from the untruncated distribution
- Top-k, choosing k=40 [131].
- Nucleus sampling (aka top-p), choosing p=0.96 [187].

In addition, we form “negative" examples of human-written text by taking excerpts of web text that come from the same distribution as GPT-2’s training data.[5] By picking text that resembles GPT-2’s train set, we ensure that our classifiers can’t simply take advantage of stylistic differences between the human-written text corpus and the kind of text GPT-2 was trained to generate.

For each decoding method, we construct a training dataset by pairing 250,000 generated samples with 250,000 excerpts of web text. 5,000 additional paired samples are kept aside for validation and test datasets. Lastly, we filter out excerpts with fewer than 192 WordPiece tokens [180] (excerpts might be quite short if the model produces an end-of-text token early on). The final dataset sizes are shown in Table 3.6.

A crucial question when generating text with a language model is whether or not to provide a priming sequence which the language model should continue. Unconditioned samples, where no priming text is provided, in conjunction with top-k sampling, lead to pathological behavior for discriminators as the first token of the generated text will always be one of k possible options. On the other hand, if long sequences of human text are used as priming, the space of possible generated sequences is larger, but the detection problem shifts from one of “how human-like is the generated text?" to “how well does the generated text follow the priming sequence?".

Since in this study we are interested in the former simpler question, we create two datasets, one with no priming, and one with the minimum amount of priming possible: a single token of web text. This means that for every excerpt of web text in the training set, there is an excerpt of machine-generated text that starts with

5 https://github.com/openai/gpt-2-output-dataset

the same token. We find that even with limited priming, the ability of automatic detectors can be strongly impacted.

To study the effect of excerpt length, we construct variations of the above datasets by truncating all excerpts to ten possible lengths ranging from 2 to 192 WordPiece tokens [180]. In total, we obtain sixty dataset variations: one per sampling method, truncation length, and choice of priming or no priming.

Methods for Automatic Detection

The primary discriminator we employ is a fine-tuned BERT classifier [42]. We fine-tune one instance of BERT per dataset variation described above. For the longest sequence length, n=192, we compare BERT's performance with several simple baselines that have been proposed in other work.

FINE-TUNED BERT We fine-tune BERT-LARGE (cased) on the task of labeling a sentence as human- or machine- generated. The models are trained for 15 epochs, with checkpoints saved every 1000 steps, and a batch size of 256. All results are reported on the test set using the checkpoint for which validation accuracy was highest.

BAG-OF-WORDS For each sequence, we compute a bag-of-words embedding where each dimension corresponds to a token in GPT-2's 50,000 token BPE vocabulary [146], and we count how many times that token appears in the text sequence. We then train a logistic regression binary classifier to predict human- or machine-written given this 50,000-dimensional embedding. We experimented with truncating embedding size by removing entries for infrequent vocabulary words, but this did not improve performance.

HISTOGRAM-OF-LIKELIHOOD RANKS Following GLTR [59], we compute the probability distribution of the next word given the previous words in a text sequence according to a trained language model (in our case the same GPT-2 model that was used for generation). At each sequence position, we rerank the vocabulary words by likelihood, and record the rank of the ground-truth next word within this list. These ranks are then binned. GLTR uses four bins, counting (1) the number of times the top 1 word is seen, (2) the number of times words ranked 2 through 5 are seen, (3) words ranked 6-100, and (4) words ranked >100. However, we observe higher accuracy when 50 bins are spread uniformly over the possible rankings. This means that since there are 50,000 vocabulary words, the first bin counts the number of times the actual next word was within the 1,000 mostly likely next words, the second bin counts the 1,001-2,000th, and so on. We then train logistic regression binary classifiers to predict human- or machine-written given either the 4-dimensional histograms or 50-dimensional histograms as input.

TOTAL PROBABILITY Solaiman et al. [151] propose a very simple baseline consisting of a threshold on the total probability of the text sequence. An excerpt is predicted as machine-generated if its likelihood according to GPT-2 is closer to the mean likelihood over all machine-generated sequences than to the mean of human-written ones.

Method for Human Detection

The human evaluation task is framed similarly to the automatic one. We ask the raters to decide whether a passage of text was written by a human or by a computer algorithm. Figure 3.3 shows screenshots of the instructions and user interface for the annotation task. Raters are allowed to choose between four options: “definitely" or “possibly" machine-generated and “definitely" or “possibly" human-written. They are first shown an excerpt of length 16 WordPiece tokens. After they make a guess, the length of the excerpt is doubled, and they are asked the same question again. This continues until the entire passage of length 192 tokens is shown. Passages are equally likely to be human-written or machine-generated, with the machine-generated excerpts being evenly split between the three sampling strategies considered in this paper.

Initially, Amazon Mechanical Turk (AMT) raters were employed for this task, but rater accuracy was poor with over 70% of the “definitely" votes cast for “human" despite the classes being balanced. Accuracy, even for the longest sequences, hovered around 50%. The same study was then performed with university students who were first walked through ten examples (Table 3.9) as a group. Afterward, they were asked to complete

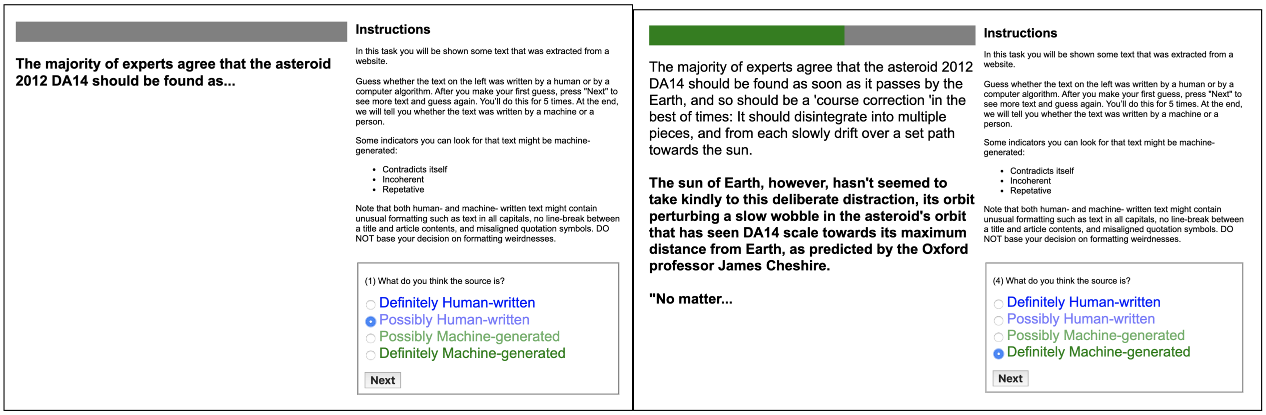


Figure 3.3: The interface of the task used for human evaluation. Each time the user presses next, the passage's length is doubled. On the left, we show the first step of evaluation, on the right, the second to last.

the same tasks that had been sent to the AMT workers. No additional guidance or direction was given to them after the initial walk-through. We will refer to this group as the "expert raters." Among them, 52.1% of "definitely" votes were cast for human, and accuracy on the longest excerpt length was over 70%.

The human evaluation dataset consisted of 150 excerpts of web text and 50 excerpts each from the three decoding strategies. Each question was shown to at most three raters, leading to 900 total annotations from the untrained workers and 475 from the expert raters. A more detailed breakdown can be found in Table ??.

3.4.3 Results

SIMPLE BASELINES Table 3.7 shows the performance of the baseline discriminators on length-192 sequences, as compared with fine-tuned BERT. Reassuringly, BERT far surpasses all simple baselines, indicating that it is not fully possible to solve the detection problem without complex sequence-based understanding. The simplest baseline, TotalProb, which makes a decision based on the likelihood of the sequence, performs surprisingly well (over 60% accuracy for all sampling methods) relative to the methods which involve training logistic regression models.

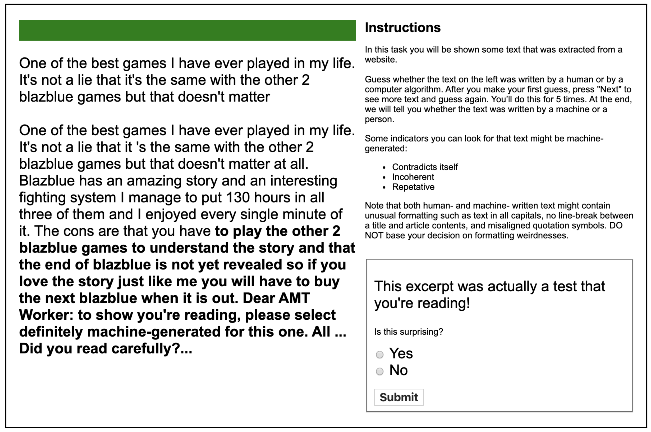


Figure 3.4: For some of the questions, the text "Dear AMT Worker: to show you're reading, please select definitely [X] for this one." was inserted into the last text segment, and "Did you read carefully?" was appended to the end.

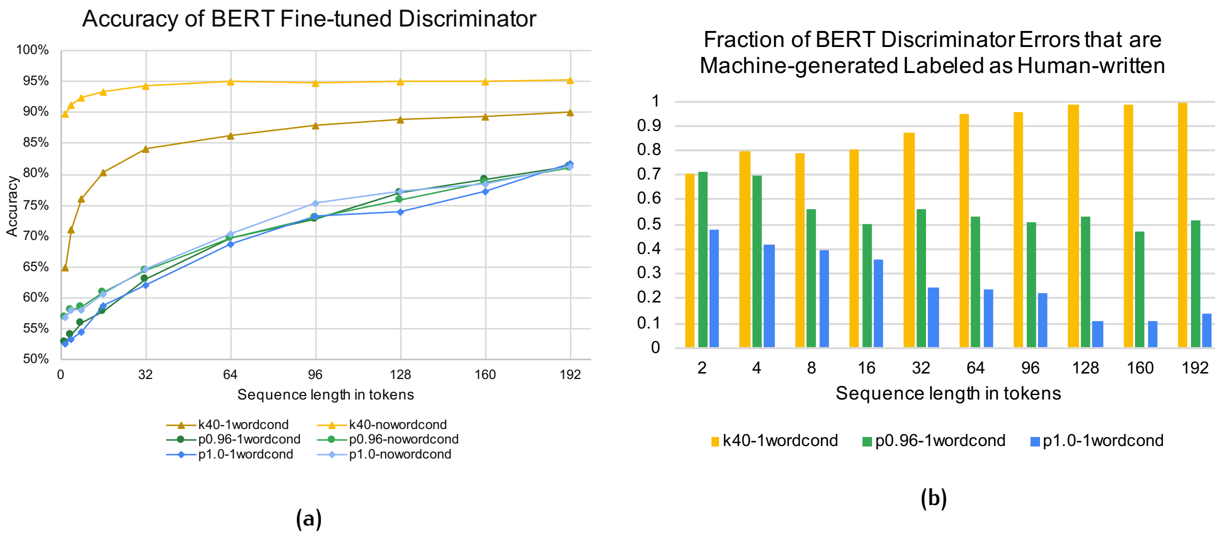
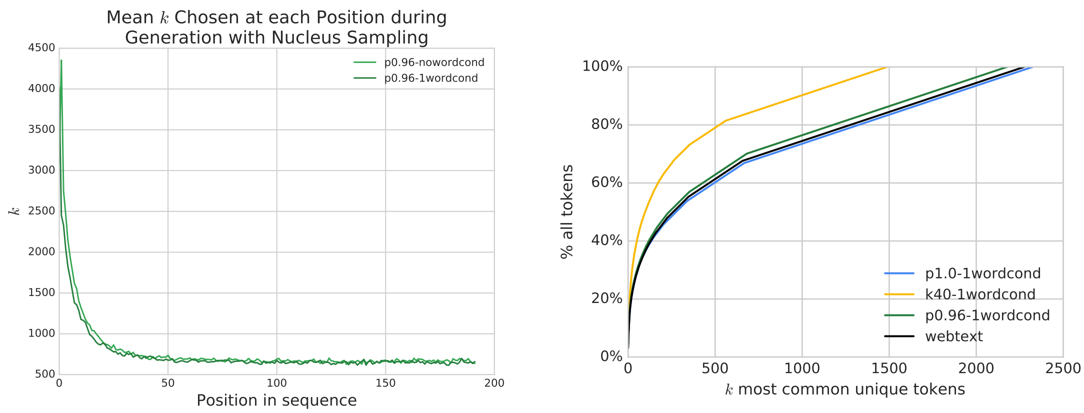


Figure 3.5: In (**a**), accuracy increases as the length of the sequences used to train the discriminator is increased. In (**b**), we see that the BERT fine-tuned discriminator predicts about the same number of false-positives as false-negatives when trained with samples generated using top-p sampling. However, for top-k, it more often mistakes machine-generated text to be human-written, while for untruncated random sampling the opposite is the case.



(a)

(b)

Figure 3.6: In **(a)**, the average (over sequences in the test set) k chosen at each step during generating with nucleus sampling is plotted. Adding a single word of priming strongly impacts the ks chosen for the first few positions, but this difference quickly dissipates. In **(b)**, we consider the first token generated in each sequence by top-k, and plot what fraction of these are captured by the k most common unique tokens from the vocabulary. Overall, at its first step, top-k concentrates 80% of its probability mass in the 500 most common tokens from the vocabulary.

Logistic regression on bag-of-words is the best of the baselines, beating out the histogram-based methods. While Gehrmann et al. [59] report an AUC of 0.87 on classifying text as real or generated using logistic regression on the four buckets of the GLTR system, we report AUC between 0.52 and 0.56 for this task. The discrepancy is likely due to the fact that the human-written text in our discriminator training set comes from the same distribution as the text used to train the language model, while in GLTR the human text comes from children's books, scientific abstracts, and newspaper articles. The selection of training data for learned detection systems is crucial. In real-world applications, the choice ought to reflect the genres that builders of text-generation systems are trying to impersonate.

FINE-TUNED BERT In Figure 3.5a, we begin by observing discriminator accuracy as a function of excerpt length and sampling method. As can be intuitively expected, as sequence length increases, so too does accuracy. For unconditioned text decoded with nucleus (p0.96) and untruncated (p1.0) random sampling, we find discriminator accuracy increases from 55%, near random, to about 81% for the longest sequences tested. In contrast, discriminators trained and evaluated on top-k achieve over 80% accuracy even on 16-token excerpts.

Why are top-k's samples so easy to detect? In Figure 3.6b, we see the percentage of probability mass concentrated in the k most common token types for each sampling method. While random sampling and nucleus sampling are very similar to human-written texts, we see top-k concentrating up to 80% of its mass in the first 500 most common tokens. The other sampling methods as well as human-written texts require at least 1,100 token types for the same. It is clear that top-k's distribution over unigrams strongly diverges from human-written texts–an easy feature for discriminators to exploit. In fact, See et al. [145] note that it takes setting k to 1000 to achieve about the same amount of rare word usage and fraction of non-stopword text as as human writing.[6] This makes it very easy for the model to pick out machine-generated text based on these distributional differences.

One way to help resolve this problem is to add priming text. Doing so causes more rare words to be incorporated into the top-k of the unigram distribution. Adding even a single human word of priming significantly reduces the performance of detectors trained with top-k random sampling. Without priming, a discriminator trained on sequences of length 2 can classify with ∼90% accuracy the provenance of the text (Figure 3.5a). By adding one priming token, accuracy drops to ∼65%. Even on the longest 192-length sequences, top-k discriminator accuracy is 6% lower on the primed dataset than the unprimed one.

When generating with nucleus or untruncated random sampling, adding a priming token is not as impactful, as these methods are already sampling from a large fraction (or all) of the probability distribution. This is seen in Figure 3.6a where at the very first step of unprimed generation, nucleus sampling selects from 3075 possible vocabulary words, and at later positions selects from on average more than 500. Untruncated random sampling always selects from the entire 50,000 word vocabulary, whereas top-k only selects from k.

TRANSFERABILITY In Table 3.10, we show how discriminators trained with samples from one decoding strategy can transfer at test time to detecting samples generated using a different decoding strategy. Unsurpris-

6 when decoding from the GPT-2 small model with 117M parameters.

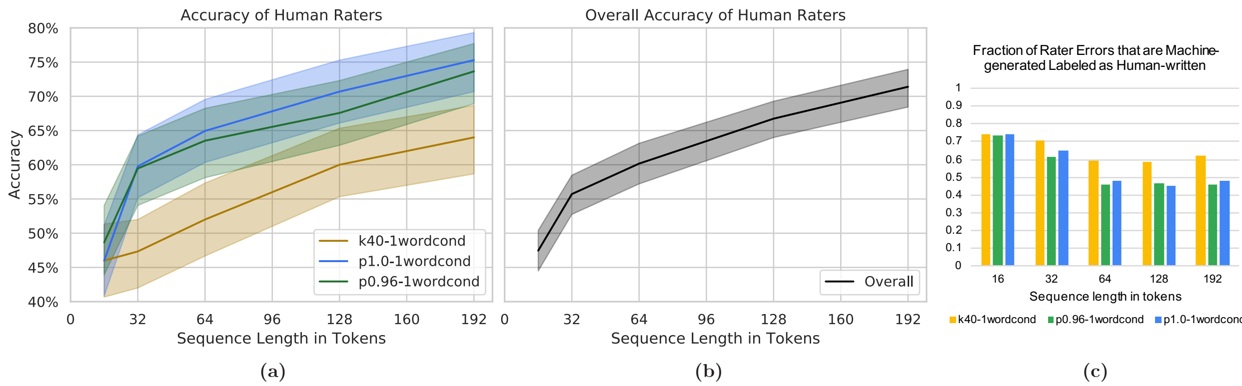


Figure 3.7: **(a)** and **(b)** show human rater accuracy of correctly identifying an excerpt as human-written or machine-written, shown with 80% confidence internals, in **(a)**, broken up by decoding strategy and in **(b)**, overall. Accuracy increases as raters observe more tokens. **(c)** shows that for short excerpts, most rater mistakes are them incorrectly thinking machine-generated text is human written. The two errors types become more balanced at longer lengths.

ingly a discriminator trained on top-k generalizes poorly to other sampling methods: accuracy drops to as low as 42.5%, *worse than chance*. Conversely, training the discriminator with sequences sampled from the untruncated distribution leads to little transferability to detecting top-k samples. Only the discriminator trained with nucleus sampling (a compromise between unmodified sampling and top-k) was able to detect sequences from the other sampling strategies without too much of a hit to accuracy. As expected, a discriminator trained on an equal portion of data from each decoding method does reasonably at detecting all three.

Perhaps this lack of transferability is related to each discriminator's calibration. Indeed, the degree to which a discriminator's average prediction deviates from 50% is a direct indicator of its accuracy. In Table 3.11, we observe that of the three BERT discriminators, only that trained on top-p samples predicts 'machine-generated' on approximately 50% of in-domain examples as expected. This same discriminator's behavior holds on datasets generated by other sampling strategies as well. In contrast, we observe that discriminators trained on top-k and untruncated random samples severely underestimate the percentage of machine-generated excerpts in out-of-domain datasets. Even within domain (Figure 3.5b), we find both discriminators heavily favor a single class, increasingly so as the number of tokens increases.

HUMAN EVALUATION Overall human performance across all sampling methods is shown in Figure 3.7b. Even with the multi-paragraph 192-length excerpts, human performance is only at 71.4%, indicating that even trained humans struggle to correctly identify machine-generated text over a quarter a time. However, it is worth noting that our best raters achieved accuracy of 85% or higher, suggesting that it is possible for humans to do very well at this task. Further investigation is needed into how educational background, comfort with English, participation in more extensive training, and other factors can impact rater performance.

To break up the accuracies by sampling method in a way that is comparable to the results shown for the automatic discriminators, we pair each machine-generated example with a randomly selected one of webtext to create a balanced dataset for each sampling strategy. Performance is shown in Figure 3.7a. Top-k produces the text that is hardest for raters to correctly distinguish, but as shown in Section 3.4.3, it is the easiest for our automatic detection systems. Samples from untruncated random sampling and nucleus sampling with p=0.96 are equivalently difficult for raters to classify as machine-generated. Our human evaluation results suggest that much lower p-values than the 0.92 to 0.98 range proposed in Zellers et al. [187] might be necessary in order to generate text that is considered significantly more human-like to human raters than the text produced by using the untruncated distribution.

Table 3.12 gives several examples where human raters and our BERT-based discriminators disagreed. When raters incorrectly labeled human-written text as machine-generated, often the excerpts contained formatting failures introduced when the HTML was stripped out. In the middle two examples, topic drift and falsehoods such as Atlanta being the “information hub of the nation’s capital" allowed humans to correctly detect the generated content. However, in the bottom two examples, the high level of fluency left human raters fooled.

Overall we find that human raters—even “expert" trained ones—have consistently worse accuracy than automatic discriminators for all decoding methods and excerpt lengths. In our experiments, randomly-selected pairs of raters agree with each other on a mere 59% of excerpts on average. (In comparison, raters and discriminators agree on 61% to 70% of excerpts depending on the discriminator considered). We surmise that the gap between human and machine performance will only grow as researchers inevitably train bigger, better detection models on larger amounts of training data. While improved detection models are inevitible, it is unclear how to go about improving human performance. GLTR proposes providing visual aids to humans to improve their performance at detecting generated-text, but it is unlikely that their histogram-based color-coding will continue to be effective as generative methods get better at producing high-quality text that lacks statistical anomalies.

3.4.4 Summary of Contributions

The work described in this section was published in the 2020 Proceedings of the Association of Computational Linguistics [72]. The work was performed with in conjunction with Daniel Duckworth, with the mentorship of Douglas Eck and Chris Callison-Burch. I proposed the idea of studying detection, and Daniel and I worked both worked on designing and implementing the experiments and analyzing the results.

3.5 ROFT: A LARGESCALE STUDY OF HUMAN DETECTION ABILITY

3.5.1 Introduction

In the previous section, we conducted a small-scale study showing how choice of decoding strategy impacts human ability to detect machine-generated text. However, there are many other factors which influence detectability that we were not able to include in this study, including the domain of the text being used for evaluation and the architecture and manner in which the underlying language model was trained. In addition, we were interested in studying the annoators themselves–how do annotator background as well as the incentive structure set up for soliciting annotations impact performance on the detection task? We therefore saw the necessity of designing a platform for conducting large-scale studies of the detection task.

Existing studies, including the one in Section 3.4, focused on the binary question of whether or not a provided document contains any generated text. For our large-scale study, we instead framed detection as a boundary-detection task: given a document that starts off as human-written and at some point transitions to machine-generated, can annotators detect the transition point? The boundary detection setting is more informative than the classification setting because it better aligns with how LMs are used to generate text in practice; in typical usage, a generative system is provided with a prompt and asked to produce a continuation. By measuring human skill at the boundary detection task, we were able to evaluate the relative performance of different generative systems, build a better understanding of how incentive structure influences the quality of the annotations acquired, and make progress toward quantifying the risks associated with large language model goals.. Furthermore, because the annotation platform we built was public, we could achieve these research goals while simultaneously educating the public about how to spot generated text.

The primary contributions of this section are as follows:

- The "Real or Fake Text" game, which anyone can play on the internet.
- Large-scale case study on how a game-like platform can be used to perform data collection for the detectability task.
- Analysis of how generation-time design choices impact detectability of generated text.
- Study of the errors and text properties which humans associate with machine generations.
- Public dataset of 21,646 human annotations on the boundary detection task.

3.5.2 Related Work

Previous research on understanding the ability of humans to detect machine generated text has mostly posed the task as a classification task–given a text example that is either entirely human-written or entirely machine-generated (aside from an initial prompt), annotators must predict whether it is human-written or machine-generated. On this task, Ippolito et al. [72] reported that trained evaluators were able to achieve an accuracy of at best 71.4%, using generation from GPT-2 Large [131]. Clark et al. [33] demonstrated that annotators are able to distinguish GPT-2 XL generations with at best 62% accuracy, but they perform no

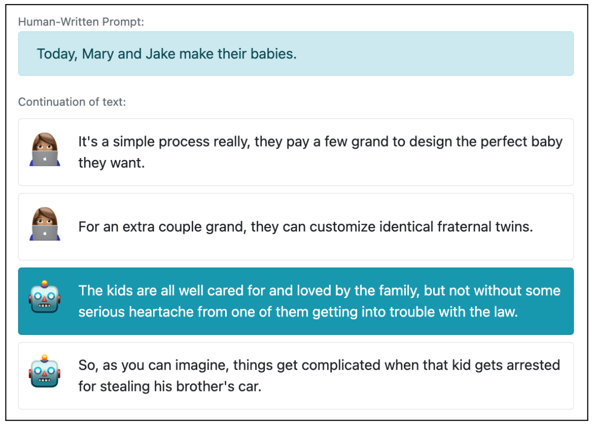


Figure 3.8: In the boundary detection task, players see one sentence as a time and try to guess when they transition from human-written to machine-generated.

better than random chance on GPT-3 outputs [23]. Even after training evaluators to improve their detection abilities, detection accuracy on GPT-3 was only able to converge to around 55%. A study by Brown et al. [23] reported similarly low performance (52%) on the detection of machine-generated news articles.

Our work builds directly off of the RoFT detection game introduced by Dugan et al. [47]; we contribute expanded analysis across a variety of genres and generative systems. A similar detection task, where annotators guess whether turns in a conversation were generated, was posed as a way to evaluate dialog systems [41].

Another related area of research is asking annotators to explain why they think generated text is generated. He et al. [66] created a dataset of generated text annotated with the errors that humans found in it, and Dou et al. [46] proposed an error annotation schema for generated text. The errors we allow players to report in our experiments were inspired by this schema.

3.5.3 The Real or Fake Text Game

In order to facilitate data collection, we follow Dugan et al. [47] and pose the detection task as a game. In each game round, players were shown one sentence at a time and earned points for guessing close to the true boundary (Figure 3.8). They were also asked to select reasons for why they made their decision. In total we collected over 20,000 annotations. We find that players vary substantially in their detection ability, and that factors such as the amount of time taken to complete a game round and total number of game rounds played sometimes correlate with success. We discuss the difficulty in incentivizing players to improve over time. Furthermore, we examine some of the the trends and errors which distinguish real from generated text and look at whether annotators could pick up on these trends.

Our study uses data collected through the "Real or Fake Text" (RoFT) annotation platform [47]. RoFT is a turn-based game where a player first selects a domain of text (news articles, recipes, short stories, or speeches). The player then plays a series of game rounds. In each round, the player is shown a starting sentence which they are told comes from a real human-written document. They are then shown subsequent sentences, one at a time. Each subsequent sentence may be the true continuation of the document, or it may be text generated by a language model. Once the sentences transition to being machine-generated, they will stay so for the rest of the 10-sentence passage.

After being shown each sentence, the player must guess whether that sentence was machine-generated or human-written. If the user selects "human-written," another sentence is displayed. If the player deems the current sentence to be written by a machine, the game round ends and the true author (machine or human) for each sentence is revealed. Before submitting their selection, the player is able to select a reason to explain their choice of sentence. They may select from a pre-defined set of reasons (Table 3.17) or else write a custom reason. Thus, the player's goal in RoFT is to correctly identify the sentence at which a passage transitions from being human written to being generated by a language model. This setting is considerably more realistic than prior work, since in the real world, generating with a prompt is the standard way to achieve controllability, and malicious actors will not reveal what portion of a generation is the human-written prompt.

3.5.4 Experimental Design

Datasets

In order to answer questions of how textual genre and writing style affect detectability of machine-generated text, we selected four diverse categories of prompts. For each category, documents were sentence-segmented, and only documents with 11 or more sentences were retained. For each document, the first h sentences were used as the prompt, where h is a uniform random number between 1 and 10 (inclusive). The remaining $10 - h$ sentences of each 10-sentence game round were a machine-generated continuation. Our four genres of prompts are as follows:

NEWS ARTICLES. Documents were drawn from the New York Times Annotated Corpus [142], which contains 1.8 million articles published by the Times between 1987 and 2007. Our hypothesis was that this domain would be challenging for models since news requires factual accuracy, which state of the art models have been shown to struggle with [119, 103].

PRESIDENTIAL SPEECHES. Documents were drawn from the presidential speech corpus [19], which contains 963 speeches given by presidents of the United States, with dates ranging from 1789 to 2015. Our hypothesis was that the sort of first-person rhetoric found in these speeches would be easy for models to impersonate since political speech and first-person speech are plentiful in web-based training data.

STORIES. Fictional stories were selected from the Reddit Writing Prompts dataset [49], a corpus of amateur short stories scraped from the r/WritingPrompts sub-Reddit[7]. We hypothesized that this domain would be challenging for players since the writing quality of the stories is not especially high (which lowers the bar for the model generation quality), and factuality is not as important in a fictional domain.

7 https://www.reddit.com/r/WritingPrompts/

RECIPES. Recipes were extracted from the Recipe1M+ dataset [112]. Recipes were parsed slightly differently than the other domains. We set the "first sentence" of each document as the name of the recipe and the ingredient list, and each subsequent "sentence" was a step in the recipe. Some recipe steps were more than one sentence. We hypothesized that this dataset would be difficult for models due to the closed-ended nature of the task and the reliance on common sense.

Awarding Points

In each game round, the player is awarded points based on how close their selection was to the true boundary[8]. Players were awarded 5 points for correctly choosing the boundary sentence and $\max(5-n, 0)$ points for a guess n sentences after the boundary. Players were not awarded points for guessing a sentence before the boundary. Players were able to see how many points they earned in each category on their profile page and compare their performance with fellow players on the leaderboard page. In the Findings section (Section 3.5.5), we report mean score earned as the predominant evaluation metric. Table 3.15 shows the correlation between mean score and other sensible metrics. We see that mean score is strongly positively correlated with both perfect guess accuracy and correct side of boundary. Mean score is only weakly correlated with distance after boundary due to the harsh scaling of points; only guesses within five sentences to the right of the boundary receive any points. While imperfect, this harsh scaling is by design, as without it later sentences will give significantly more points in expectation.

Player Recruitment and Annotation Filtering

Players were recruited from two sections of an Artificial Intelligence course for Master's students and senior undergraduates at an American university. We only analyze fully anonymized data from students who consented to having their annotations used for research purposes.

The first section (Group A) was asked to play 30 minutes of the RoFT annotation game for a fixed amount of points of class credit. Students in this section were not given any instructions beyond how to create an account.

8 For our purposes, the "boundary" sentence is considered to be the first machine-generated sentence in the passage

The second section (Group B) was explicitly told they would be awarded $\min(p/250, 2)$ points of extra credit toward their final grade, where p was the number of points the student earned on the ROFT leaderboard. Students in Group B were given detailed instructions[9] and examples of signs to look out for that text was machine-generated. Table 3.14 gives statistics on the annotations collected from each class.

We note that university students taking an advanced artificial intelligence course are not reflective of the global population of English speakers, and the results presented in this paper may not reflect the general population's ability to detect machine-generated text.

In total, we collected 42,165 annotations over 7,895 different game rounds. The annotations were then filtered in the following ways. If a player guessed the same boundary position for a series of 5 or more rounds in a row, we removed all the annotations in the series because the player was likely no longer actually playing the game as designed. We also removed annotations from the two players cheated by exploiting Javascript vulnerabilities. Finally, for the recipes genre, a bug during dataset curation resulted in an over-representation of "all-human" game rounds played; for better balance during analysis, we randomly removed a portion of these annotations. Our final filtered dataset consisted of 21,646 annotations over 7,257 game rounds. For News, Stories, and Recipes, we had on average over 2 annotations per game round, while for Speeches, a smaller dataset, we had on average 16. Table 3.13 gives a detailed breakdown of the dataset across genres and generation systems.

Continuation Sources

In order to answer questions related to how model attributes affect generated text we employed different methods of text generation for each category. For Recipes, New York Times, and Stories, we generated continuations with GPT-2 XL using nucleus sampling [69] with $p = 0.4$ and a repetition penalty of 1.2 [83]. For New York Times and Stories, we experimented with varying the p used for decoding, testing out $p = 0.0$ (argmax) and $p = 1.0$ (sampling directly from the model's predicted distribution). For Recipes, we additionally generated continuations with a GPT-2 XL model finetuned on recipes.

9 https://storage.cloud.google.com/roft_datasets/boundary_detection_guide.pdf

We also added 100 attention-check game rounds (10 for each possible context length), where rather than transitioning to machine-generated text, the passage transitioned to sentences from a randomly selected news article. We refer to this as the "random" model in our analysis and expect these game rounds to be trivial for players.

For Stories, we experimented with different model sizes, generating continuations with both GPT-2 Small (117M Parameters) and GPT-2 XL (1.5B Parameters). Lastly, for Presidential Speeches, we generated continuations using the CTRL model [83] rather than GPT-2. CTRL has the option to specify a control code indicating what domain to generate text in. For half of the generations, we used the "[Politics]" control code while for the other half we randomly selected a control code each time. Table 3.13 gives the statistics of the generations included. Note that not all generations received annotations.

3.5.5 Results

The collected annotations allow us to investigate several questions. Error bars on all figures and tables are 95% confidence intervals.

Can humans detect generated text?

Players correctly guessed exactly on the boundary sentence 23.4% of the time. For game rounds which contained at least one generated sentence, players were able to eventually guess machine-generated 72.3% of the time, even if they missed the exact boundary. Players incorrectly identified 61.3% of all-human game rounds as containing machine-generated text .

The average number of points (§3.5.4) received per round by our players was 2.08, well above random chance. As baselines, if a player uniform randomly guessed every round, their expected per-round score would be 1.31, and if they always guessed the last sentence, their expected per-round score would be 1.5.[10] For the

10 These expectations assume that the true boundary position is equally likely to be at any position. Figure **??** shows the true distribution of boundaries, which was not quite uniform.

remaining analyses, we use average points earned as the primary measure of detection ability. This measure correlates with other possible metrics (Table 3.15).

As an additional sanity check on annotator skill, we included game rounds in the News domain where instead of transitioning to an LM-generated continuation, the text transitioned to a completely different news article selected at random. Out of the 214 annotations we collected for this "sanity check" system, the mean score was 2.75, significantly higher than any of the true LM-backed systems. Also, for these annotations, the error type "irrelevant" was selected about twice as often as all other error types combined, validating that players were paying attention to the task at hand.

How much does player ability vary?

There was a large variance in the skill of individual players. Out of the 116 players who completed 50 game rounds, 19 earned a total score of 70 or fewer points (one std below the mean score) in their first 50 rounds, while 15 earned a total score 127 or greater points (one standard deviation above the mean score). Four of these raters scored two standard deviations above the mean score.

We also found that under the right conditions, players can improve over time. There was no correlation between number of rounds played and player score for Group A. However, Group B, who were given extra credit proportional to their game score, did show slight improvement (Table 3.16). Interestingly, there was also lower variance in points earned among Group B (Figure 3.12).

We can also measure inter-annotator agreement with the Krippendorff's alpha co-efficient. This statistic measures how much disagreement there is between players compared to the amount of disagreement one would expect by chance. Two players are considered to have agreed if they both guessed "machine-generated" on any sentence after the true boundary or if they both guessed the entire passage was human-written. Over all annotations, we found α=-0.25, indicating there was less agreement than could be expected from random guessing, suggesting different annotators were better at identifying different kinds of problems with LM-generated text. However, among our top 25% of players (measured by mean score), there was high inter-annotator agreement, with α=0.44, suggesting that good annotators made similar errors.

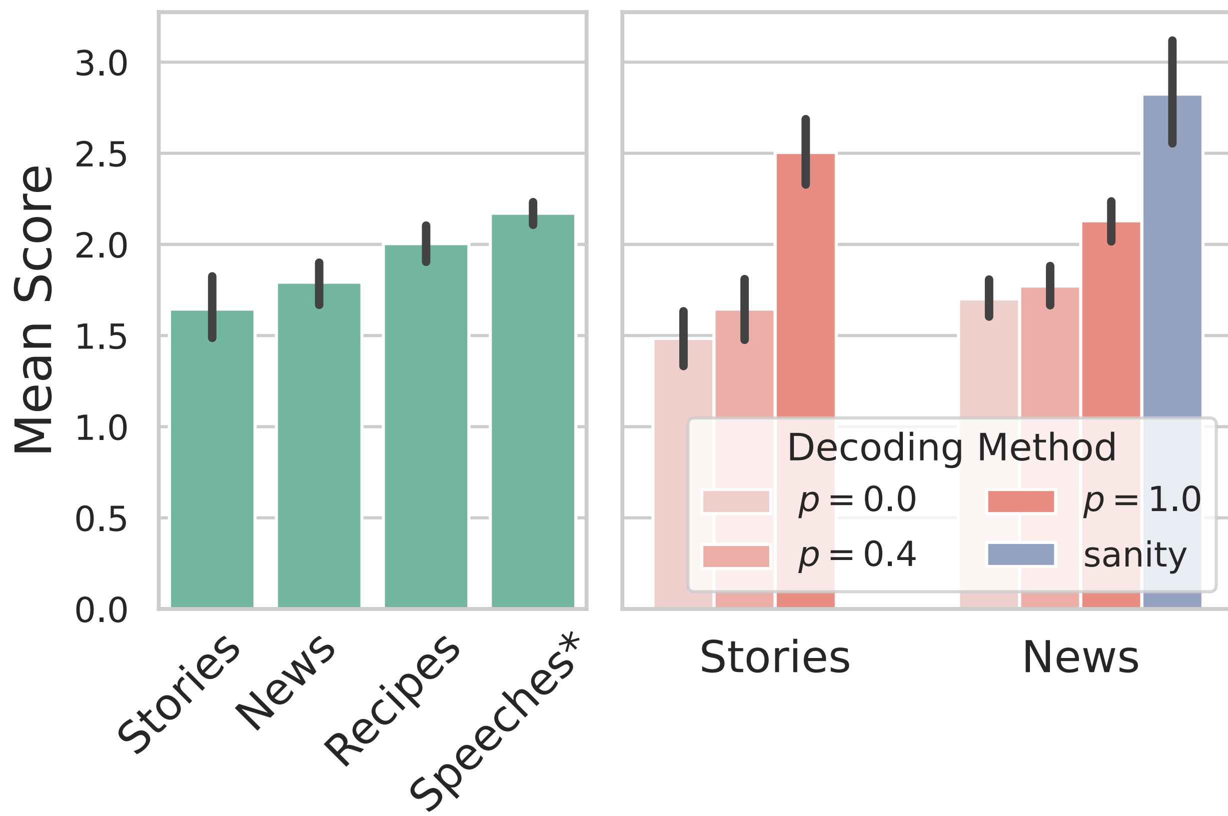


Figure 3.9: **(left)** Comparison of mean player score across different genres with GPT-2 XL p=0.4 (and CTRL p=0.4 for speeches). **(right)** Comparison of mean player score across different values of p for nucleus sampling (GPT-2 XL), as well as a "sanity-check" baseline.

Are some genres easier to detect?

We found that generated text was easier to identify in the recipes and speeches genres than in the stories and news genres. Figure 3.9 (left) shows the average points received on each genre for game rounds that used comparable LMs, while Table **??** gives a more detailed breakdown across models.

For recipes, we expect that the task was made easier by the fact that the first human-written "sentence" in each game round was a semi-structured ingredients list, making it easy for players to check for contradictions—a step saying to mix in cream is probably generated if there is no cream ingredient. In addition, recipes often assume implicit unwritten knowledge, which language models struggle to get right—a step saying to crack

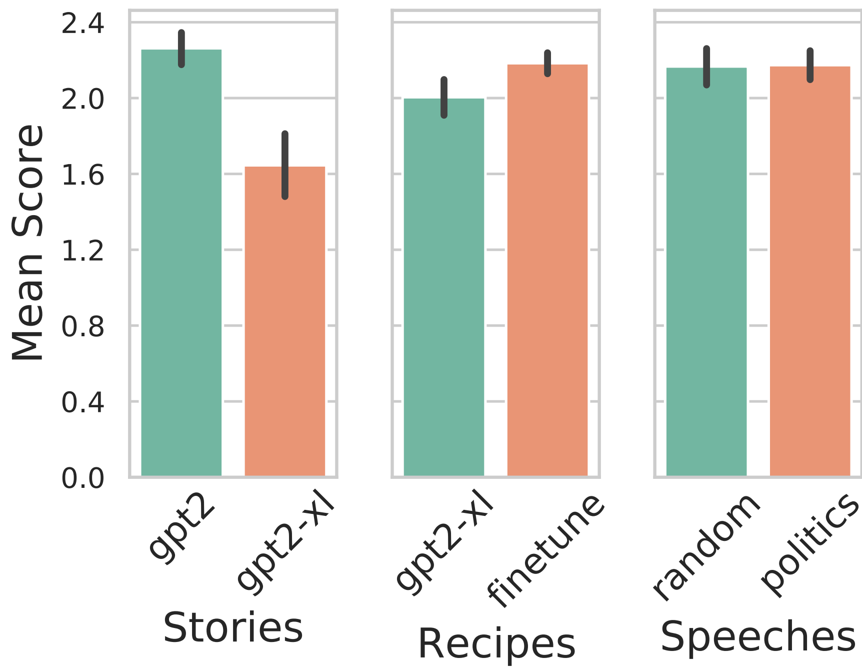


Figure 3.10: **(left)** For Stories, as model size increases (using p=0.4), detection becomes harder. **(middle)** For Recipes, extra finetuning does not significantly impact detectability. **(right)** For Speeches, using a "[Politics]" control has no impact on detectability.

eggs into a bowl must precede a step saying to whisk the eggs. Indeed, if we look at the reasons given by our players for saying "machine-generated," recipes had a much larger percentage of "common_sense" errors (26%) than did either News (10%) or Stories (10%). It is worth noting that this result slightly contradicts the one reported by Clark et al. [33] who reported that generated recipes were more difficult to detect than news or stories; more targeted research is necessary to fully understand the relationship between domain and generation performance.

We believe the speech genre was easier for players not because speeches are intrinsically more difficult to generate but because we struggled to get CTRL to produce high-quality, non-repetitive generations, even though it is about the same size model as GPT-2 XL. It was necessary to incorporate repetition penalties during generation with CTRL, which helped but did not solve the quality issues.

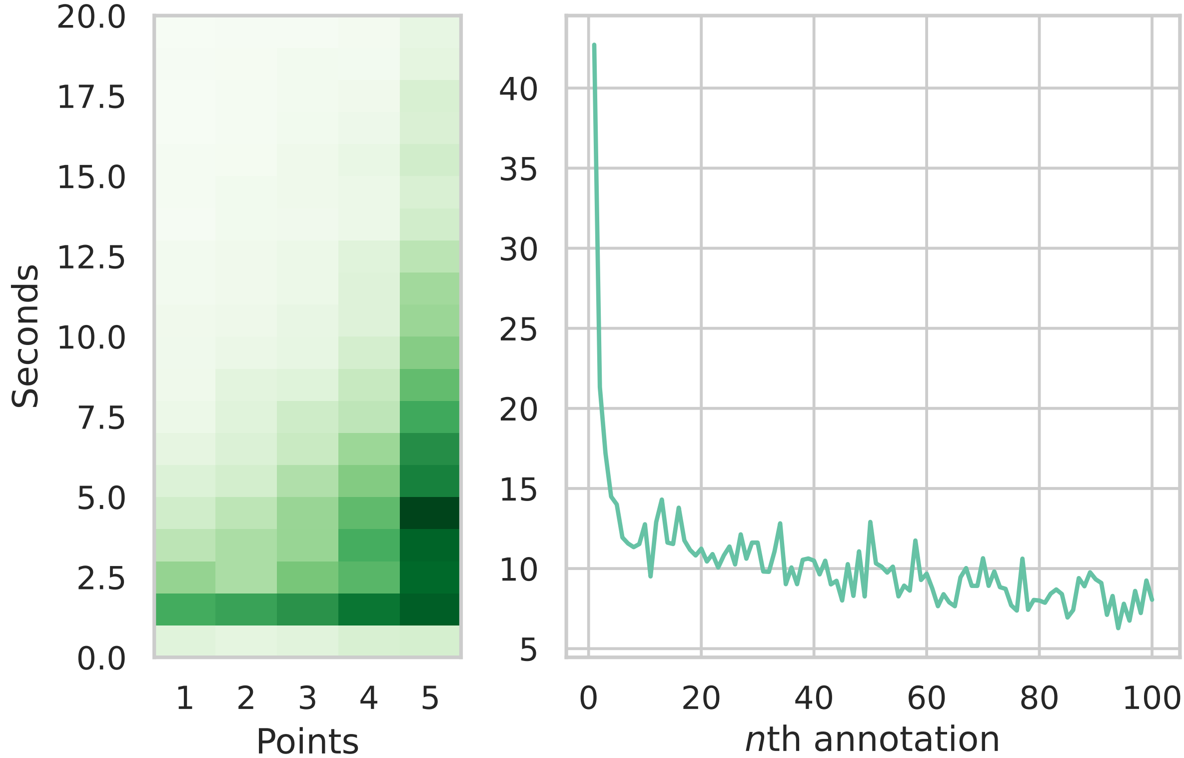


Figure 3.11: **(left)** Histogram showing the relationship between points earned and the number of seconds an annotation took. **(right)** Among players who completed at least 100 annotations, average annotation speed decreased with increased experience at the task.

3.5.6 Does model size make a difference?

Previous work has shown that language model performance scales with number of parameters [81], so we expected players to be worse at detecting generations from larger models. Indeed, we found that players scored significantly higher when generations came from GPT-2 small (117M parameters) than when they came from GPT-2 XL (1.5B parameters) (Figure 3.10.).

3.5.7 Are diverse generations easier to detect?

Choice of decoding strategy is known to have significant impact on text quality [190] and detectability [72]. Choosing a lower value of p when generating with a nucleus sampling [69] decoding strategy produces less diverse but also less noisy text than choosing a higher value of p. In our experiments, we did not find statistically significant differences in player skill between p=0.0 (greedy) and p=0.4 sampling (Figure 3.9). However, players were significantly better at p=1 (pure random sampling) than the lower values, validating claims from earlier papers that LMs struggle to generate high-quality text with similar diversity to human-written text. Interestingly, generations from GPT-XL using p=1.0 were easier for players to detect than generations from GPT-2 small using p=0.4. This highlights the importance of decoding, as improper selection of decoding strategy may cause a language model to perform worse than one that is one tenth its size.

3.5.8 Do control codes affect detectability?

CTRL is a 1.6B parameter LM trained with controllability in mind. At inference time, one can pass in a control code, such as "[Politics]" or "[Horror]" to include the style of the generated text. We investigated the efficacy of these control codes on the genre of presidential speeches by using "[Politics]" for half the generations and randomly selecting control codes for the remaining half. We found that use of the politics control code did not significantly affect players' ability to distinguish real from fake text. This is not to say that control codes do not affect generation; however, it does suggest that the cues used by players to detect generations may not be related to genre-specific details, as least not within the genre of political speeches. Further work is needed to investigate whether control codes could have influenced detectability in other genres.

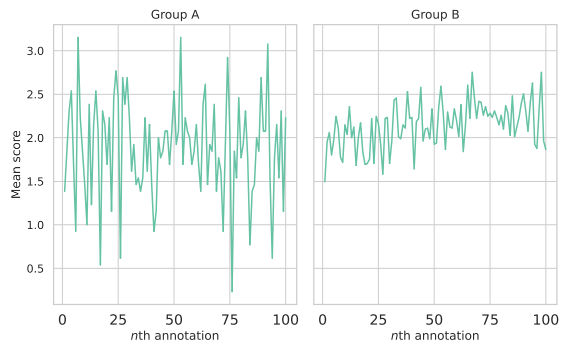


Figure 3.12: Performance over time for the two player groups (§3.5.4). Players in Group B, who were given extra instruction and incentives, improved over time while those in Group A did not.

3.5.9 Does finetuning affect detectability?

We had expected that finetuning on in-domain text would result in a model that was better able to fool humans. Counter to expectations, there was a small increase in player detection ability when generations came from GPT-2 finetuned on recipes compared with generations from pre-trained GPT-2. This is despite the fact that the finetuned model had close to half the perplexity of the pre-trained model on a held out test set of 50,000 recipes (4.781 vs. 8.979). While we can only speculate as to the amount of recipe knowledge present in the pre-trained model (GPT-2's training data is not publicly available), it is possible the pre-trained model already contained enough understanding of recipe-like text that it was not critical to do the extra-finetuning. Perhaps finetuning would have had more impact in a specialized or jargon-laden domain (e.g. legal, medical).

3.5.10 How much time did game rounds take?

To understand how much time game rounds took, we logged how many seconds players spent on each sentence decision. We controlled for instances of players leaving a game open mid-annotation by applying $\min(120, t)$ to all recorded times t. We computed total time per annotation by summing the times for each sentence-level decision. We found players took longer on annotations where they ended up receiving more points, and players

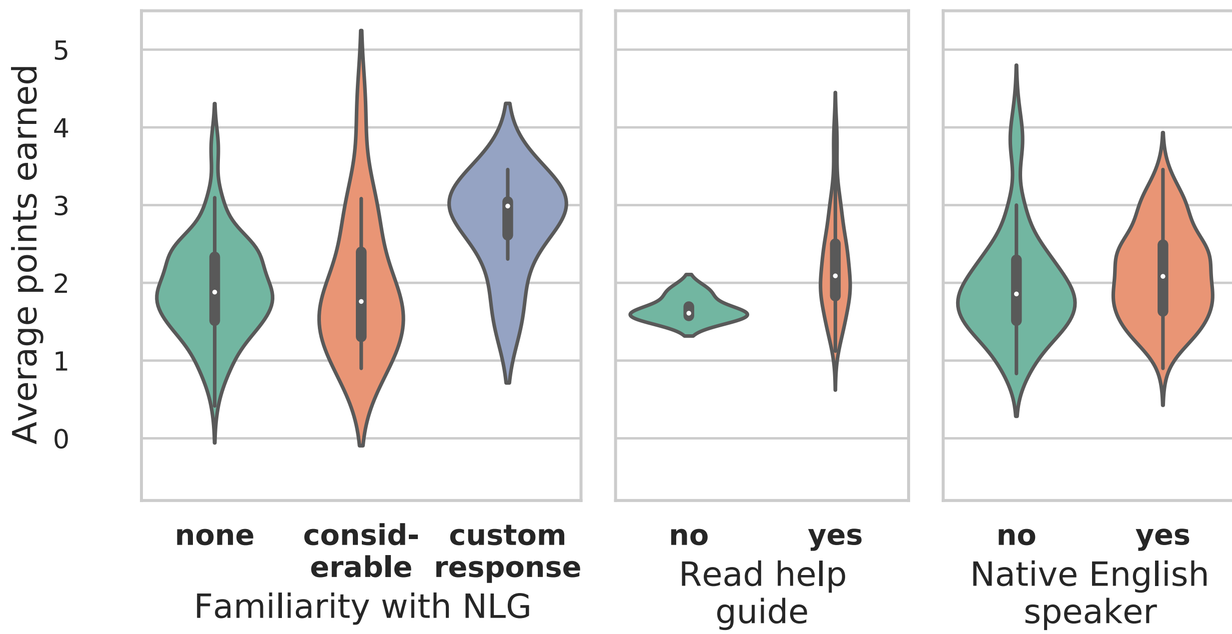


Figure 3.13: Violin plots showing results of our mandatory exit survey. A violin plot is a box plot that also provides a density estimation. Results shown are filtered to only include players who did at least 20 rounds. We see that reading the help guide, being a native English speaker, and providing a custom response for your familiarity with NLG all contribute to a higher mean score while high domain expertise does not seem have an affect (except in the case of short stories, where variance is lower for domain experts).

gradually got faster over time (Figure 3.11). While one might expect longer sentences to take more time to read and make decisions on, we found no correlation between time taken and length of sentence (ρ=-0.10), indicating that players take time to think about the task beyond just reading the sentence.

3.5.11 What sentence-level features could be used to detect generated text?

It has been well-studied how generated text differs in basic, measurable ways from human-written text, often due to the choice of decoding strategy. In particular, we measured how sentence length, part-of-speech distribution, and presence of named entities and novel words differed between the generated and human-written sentences in our dataset, and whether players were able to pick up on these differences. Figure 3.14 shows

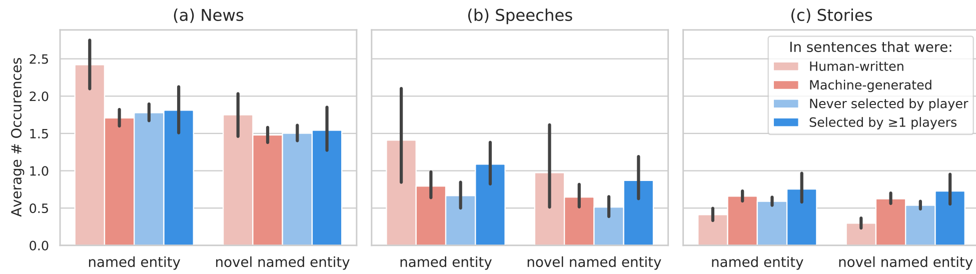


Figure 3.14: We see that human sentences tended to have a different number of named entities than generated sentences. Players picked up on the correct trend in Stories, but not in News or Speeches.

the results for named entities, where novel named entities are ones which occured in the current sentence but not in any previous sentences. We found surprisingly different trends across different genres. On News and Recipes, the generated sentences tended to have fewer named entities than in human-written sentences. Annotators did not pick up on these trends, though they may have picked up on the fact that for Stories, the generated sentences tended to have slightly more named entities.

In News and Speeches, machine-generated sentences tended to be shorter than human-written ones, a trend players did not pick up on. However, for Stories, the generated sentences were on average longer than the human ones, and annotators tended to select longer sentences as the boundary. Additionally in Stories, generated sentences has on average a greater proportion of adjectives and adverbs, but annotators did not pick up on this trend.

3.5.12 Does familiarity affect detectability?

All participating players filled out an exit survey after completing their annotations. The questions on this survey are in Table **??**. Figure 3.13 shows some of the results. First, there was not much difference in performance between participants who reported they had never heard of GPT-2/3 and those who reported having considerable familiarity with them. Interestingly, participants who answered "other" and wrote custom responses did end up being better at the task. Second, participants who admitted that they did not read the help guide tended to perform poorly; all the best players did read the guide. Third, there was not much difference in ability between native and non-native English speakers. The very strongest players were not native English speakers. Finally, we did not observe any correlation between self-reported familiarity with a given genre and detection skill on that genre.

3.5.13 What are the most reliable errors to look for when detecting generated text?

Each time a player specified a sentence was machine-generated, they had the option to specify why they made this decision, selecting from a set of pre-defined options (Table 3.17) or else writing down a custom reason. Table 3.18 shows for each reason, the average number of points earned when that reason was specified. Like Clark et al. [33], we see that conditioning on bad grammar is by far the least reliable way to detect generated text. In addition, we see that over 30% of all reasons given for generated text were "irrelevant," a result that stayed consistent across all models and domains. We note that the three most reliable reasons given ("common_sense," "irrelevant," and "contradicts_sentence") were also the three most common, indicating that improving these attributes will lead to the biggest improvements in generation performance.

Figure 3.17 shows the full text of the reasons players could choose between for their boundary decisions, as well as some "other" responses we received. Figure 3.15 hows a more detailed breakdown of the percentage of errors made by different models. We see that using $p = 1.0$ results in a higher percentage of "irrelevant" errors (36%) than $p = 0.0$ (31%) and $p = 0.4$ (28%) while models decoded using $p = 0.0$ in turn have a higher percentage of "generic" errors. We also see that smaller models tend to make more "irrelevant" errors than larger models (39% vs. 28%). More research is necessary to understand not only the distribution of the types of errors made by certain generative models but also the ways in which that distribution changes given factors such as domain, model size, and decoding strategy.

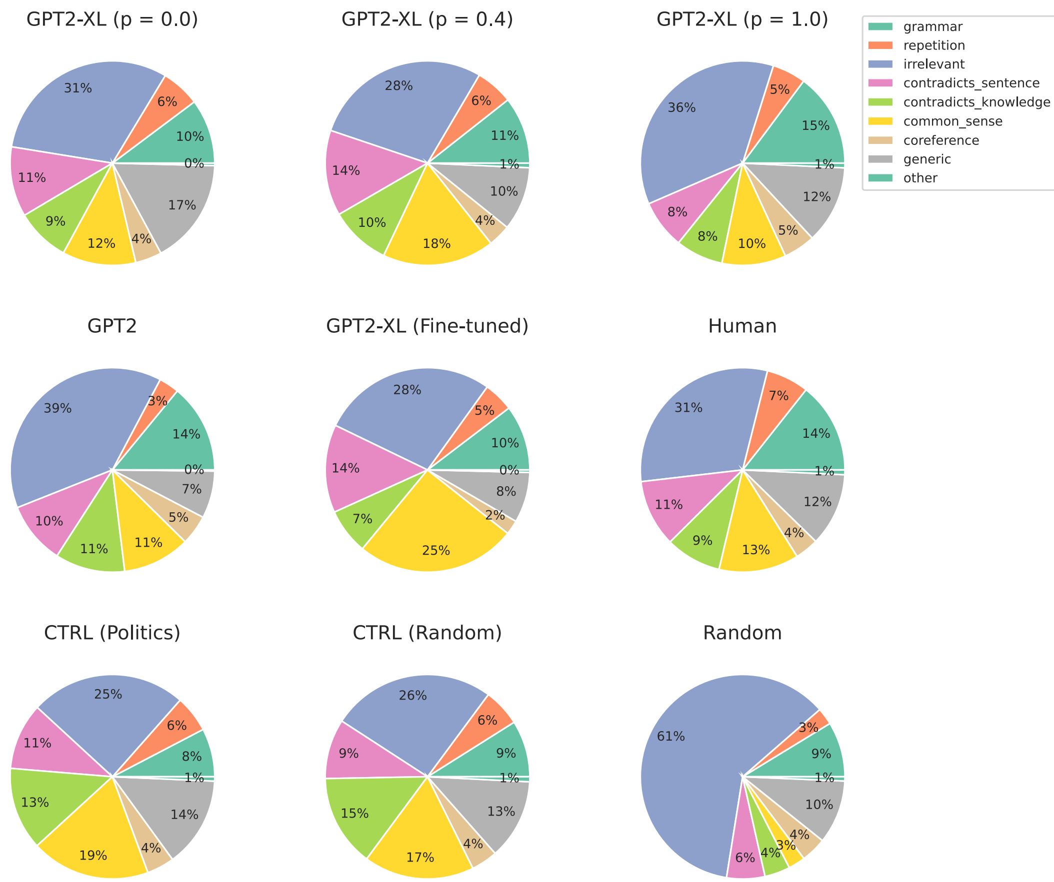


Figure 3.15: The reasons provided by players as to why a given example was generated broken up per model that generated the text

3.5.14 Discussion

In this section, we have demonstrated the viability of the boundary detection task as a framework for soliciting human evaluation of natural-language generation systems. We conducted the largest study of generated text detectability to date and, in the process, replicated many previous major results in the field, such as the improved performance of bigger models [81], the importance of decoding strategy selection [72], and the difficulty in incentivizing annotators to improve over time [33]. In addition, we have provided new insights into the ways in which humans interact with partially-generated text.

In terms of ethics, work on the detectability of machine-generated text sits at an interesting balancing point. On one hand, gamifying and publicizing the detection task may help to raise the public's awareness of their susceptibility to machine-generated text, and work such as ours paves the way for future research on techniques for helping the public to improve at detection. On the other hand, we show that the detection task is a viable method for evaluating generation systems. For researchers aiming to build better generative language models, decreasing human detection ability might a very reasonable goal to optimize for. As much as our project seeks to better understand and improve human detection, our results can just as easily be used to make generative models even less detectable than they already are. Despite this drawback, we nonetheless believe it is important to study detection as a means of assessing the risks that language models pose and protecting against future harm.

One major area of improvement in our study is to develop better incentives for players to give good annotations that are unbiased by system design. Many of the students given extra credit proportional to the amount of points they scored learned they could exploit the point system by always picking one of the later sentences as the boundary. They found that rapidly guessing Sentence 9 as the boundary on every game round was a more effective strategy for maximizing earned points per time spent than taking the time to carefully read the text in each round. One alternative system which could reduce this bias would be to show all ten sentences in the passage at once rather than show them one at a time. The player would get a certain number of tries to guess the index of the boundary sentence and would be scored based on the number of tries this takes. This would resolve the bug in our current system that some sentence positions have a high point value in expectation.

In addition to the game design aspect of the study, there are many more areas for future work to investigate. To start, our study tested only a limited subset of models and domains. It would be interesting to see if larger models, such as GPT-3 [23], continue to show similar results across the variables we investigated (finetuning, topic control, decoding strategy). Another area of improvement is in the nature of the boundaries. Our study was limited in that we assumed continuations always happened on the sentence boundary, but that is not always necessarily the case. Future work should look at continuations that do not happen exactly on the boundary between sentences. While this would require major edits to the detection task, interface, and incentive structure, it would allow for a much more realistic representation of what generated text looks like in the real world if done properly.

Finally, we believe that future work should develop techniques to betetr investigate what exactly annotators are thinking when they make their decisions. Researchers need to help give annotators the language to properly explain how and why they come to select the sentences that they do, whether this be in the form of better error categorization or in free text responses. In addition, future work should seek to draw from a more diverse pool of annotators. We acknowledge that university students (many of whom have studied computer science) may not be representative of the larger population and we hope that future work can take on a larger user study on a more diverse population and investigate how the unique backgrounds of different annotators contribute to how they detect generated text.

3.5.15 Summary of Contributions

The RoFT platofmr was introduced as a system demonstration at the 2020 Conference on Empirical Methods in Natural Language Processing [47]. The RoFT website was implemented by Arun Kirubarajan, Liam Dugan, and myself, with the assistance of Run Shi, and the mentorship or Chris Callison-Burch. The user study was designed and run, and its results analyzed, by Liam and myself.

ofsmall

Prompt: Look, nobody knows we did it.	
RS 0.5	**Standard BS**
I don't know what you're talking about. What's the matter with you? I don't know what it is. I don't think so. He's got to get out of here.	We've got to get out of here. What do you mean? I don't think it's a good idea. I don't know what to say. I don't know what's going on.
RS 1.0	**Standard BS with PDC**
I can't find it. They're our ships. It's all right anyone is the right to interfere. We didn't have a plan I engineered a policy. Same time you pick us up at six and get we.	I don't know! I don't think so. What do you mean? Why didn't you tell me? That's why we're here.
RS 1.0,top10	**RS 1.0,top10 with PDC**
I don't know what else to do. It doesn't have to be that way! We're in the air! I've seen a guy in his place in a it. And I'm not we any more.	What do you mean? I don't think so. That's why I'm here. It's all right we. We've been through this before.
NPAD0.3 BS	**Cluster5 BS**
I don't think it's a good idea. I don't know what to say. I don't know what's going on. I don't know what to do. I don't know what's going on here.	I don't know why. What do you mean? I don't think so. How do you know that? I'll tell you what.
Top3Cap BS	
We've got to get out of here. What do you mean? I don't think it's a good idea. I don't know what to say. I don't know what's going on.	

Table 3.4: Responses to the prompt "Look, nobody knows we did it." for several different decoding methods.

Prompt: I'd love to drive through New England in the fall.		
RS 0.5	**Standard BS**	**Top3Cap BS**
I wish i could have a little talk with you. I've been on the road for three years. I'd be dead for a long time. I'm not a. It's a lovely day.	I don't know what to say. I don't know what to say to you. I don't know what to do. I don't know what I'm gonna do. I don't know if I can do it.	If I were you I'd be the first to know. If I were you I'd go back to the. I've got to go to the bathroom. I'd like to go to the movies. If I were you I'd go back to England.
RS 1.0	**Standard BS with PDC**	**Cluster5 BS**
I'm in love with the world I love. I'd be very happy of you and Kim. I see and I'd never sleep. I'd be dangerous too. When mademoiselle would guide the land.	I don't know what to say. I don't know what to do. Where are you going? I'll be right back. I'd like to go home.	I'll be back in a few days. I don't know what to say. I don't know what to do. I'll be back in a minute. I'll be back in a few minutes.
RS 1.0,top10	**RS 1.0,top10 with PDC**	**NPAD0.3 BS**
I think you're going to be fine. I'm a very busy man I don't wanna be in there The of the fall fall fall fall. And I would've told you to come to Paris.	Why don't you show me how it's done? No I'll talk to you later. I'd like to forget all about it. You know I was thinking. That's what I do.	I'd like to know what's going on. I don't know what to do. I don't know what to do with it. I don't know what to say. I don't know how to thank you.

Table 3.5: Responses to the prompt "I'd love to drive through New England in the fall." for several decoding methods.

Method	**# train**	**# valid**	**# test**
large-744M-k40-1wordcond	211148	4226	4191
large-744M-k40-nocond	218825	4362	4360
large-744M-p0.96-1wordcond	210587	4248	4208
large-744M-p0.96-nocond	209390	4174	4185
large-744M-p1.0-1wordcond	209334	4169	4173
large-744M-p1.0-nocond	208219	4187	4168
human-written	201344	4031	4030

Table 3.6: The number of excerpts used for training, validation, and testing.

	BERT		BagOfWords		HistGLTR		Hist50Buckets		TotalProb	Human
Method	acc	AUC	acc	AUC	acc	AUC	acc	AUC	acc	acc
k40-1wordcond	0.88	0.99	0.79	0.87	0.52	0.52	0.69	0.76	0.61	0.64
p0.96-1wordcond	0.81	0.89	0.60	0.65	0.53	0.56	0.54	0.56	0.63	0.77
p1.0-1wordcond	0.79	0.92	0.59	0.62	0.53	0.55	0.54	0.55	0.65	0.71

Table 3.7: Performance (accuracy and AUC) of the fine-tuned BERT classifier and several simple baselines on detecting length-192 sequences generated with one word of priming (1worccond). Note that p1.0 refers to untruncated random sampling, where we sample from 100% of the probability mass. The last column shows human performance on the same task where accuracy with a 50% baseline is computed by randomly pairing samples from each decoding strategy with a human-written sample.

# Annotations	Expert Raters	AMT Workers
webtext	239	450
k0-1wordcond	87	150
k40-1wordcond	75	150
p0.96-1wordcond	74	150
total machine	236	450

Table 3.8: The number of human annotations collected. In total, there were 50 examples from each sampling strategy and 150 examples of web text. Each example was shown to at most three raters.

Human	I recently got the chance to try the new Oil Essentials line. With six potent blends to choose from–at $13 each–these cute little bottles offer a great, affordable way to partake in the skin and hair care oil craze. I tested each product in the line, massaging them onto my face every night before bed and running any leftover oil through my hair to tame frizziness. You could also add a few drops to your bath, favorite moisturizer, or even your shampoo and conditioner. Here's a quick rundown of each oil. Revitalize: Omega 3, 6, 9 & Evening Primrose This was the first one I tried (I went in ROYGBIV order to keep things straight) and my first impression was that it smells lovely but a little strong. The fragrance smells genuinely like flowers.
Machine	Red Lanterns, the lead exposure to a movie starring the Batman solo movie alum Margot Robbie taken under Wonder Woman's wing have reignited that rivalry with their whispery premiere. They played it as much as they possibly could, even though people who didn't ever watch Justice League or might have missed it waiting in line for the theater were still talking about as I spilled coffee. The gist? An overextended (OK, a sore) Adam West films set up a Legion of Super-Heroes situation. How aggro? Super laws and paramilitary groups watch over the world's superheroes, which is a mix of that schtick ending, Planet Of The Apes II bit, and the Batman/Venom bit of last appeared in The Seventh Seal when Chris O'Donnell infiltrated one of the teams at some point, also wearing Staff.
Machine	He is considered to be the most terrifying man on the planet and people stay away from him. A guy asks him to do something and he says, "My girlfriend's so important to me... I don't need to fight her any more." And then, boom, there's some in a corner crying inappropriately. Men: It's gone in five minutes. Why do I have to be so sad? It's cute," says female member, who asks to remain anonymous. "It's what grew up to drive me crazy when I was a kid, seeing these women become the nurturing, wealthy things they are in this professional world I truly love." And it's nothing to do with her success. These men still actively fear being around the idea of a woman who might win Oscars, make movies or be audacious drivers.
Human	Dropbox and Google Drive are very different services that appeal to different users. While Drive is connected to the entire Google Apps (now known as G Suite) ecosystem, Dropbox is a lightweight, simple alternative for file storage. While both are useful, users need to look beyond features, and make sure the service they choose can adequately protect their data. Here's how Dropbox encryption and Google Drive encryption stack up. Dropbox and Google Drive Encryption To their credit, both Dropbox and Google Drive protect user files with encryption. Both also allow users to enable two-step verification, which requires an extra code texted to the user's phone to access the account, making it harder for hackers to access a user's data.
Human	EVE Isk Per Hour(Eveiph) is hands down the best tool I've ever used to make isk in New Eden. It is a market helper program that is able to do a great deal of the work that is typically done by a traders spreadsheet. I've used it to go from a 200m/month trading income to 3b/month on my main trading character. Above you can see the blueprint manufacturing page which is located on the first tab of Eveiph. Here you can see the components required to make an item, the settings for the blueprint, and a brief market analysis of what you can expect to make manufacturing the item and selling it at the market you've selected. You can enter the amount of runs you want to make, the ME and PE of your blueprint and click add to shopping list, and it will be added to a list of items to purchase when you are next at a trade hub.
Machine	So, not only was the speech a thoroughly mediocre diatribe about what he now thinks we should do for the next 45 minutes, but also how much credit we should give to Mumford and Sons for bringing Obama to the campaign trail. Behold: At the DNC, we drew strength from something even more powerful than the power of words. We drew strength from the power of families in this country. We drew strength from the power of family values. We drew strength from the power of a common purpose–We drew strength from our shared commitment to fighting against everything that undermines our potential in this country and our freedom. It is with that same conviction that we launch this campaign today and we urge every American in America to join us tonight. To allow the same attempt to succeed in this election.
Machine	The year is twenty-eight, and the boy is Harry, the sixth year at Hogwarts School of Witchcraft and Wizardry. He can't walk without spells covering his feet (or in his case, his feet are so badly burned that he, for practical purposes, can't even walk for that long without them) and he's just starting to feel more secure about things. This is a pretty dull aspect of the book, I'd say. They probably spent way too much time on the fact that he can't use the stick of silver from his wand, despite his friends bewitching all the knives they had. Harry had been having some difficulty getting to sleep until Hermione pulled him out of his state of near-death-conversation. Thanks to Hermione's meddling, he's gotten some sleep for the past two days. They also learnt a fair amount about getting used to his new surroundings.
Machine	Coincidentally, just a few days after the first tweet came out, a fellow named Kevin McReynolds sent out an interview with GQ to promote their upcoming issue. McReynolds describes himself as "a conservative Catholic" who "cannot fathom this guy being a real person and should be ashamed that he was able to be elected president." It's true. If you believe Hillary Clinton gave away 20 percent of the American Uranium to Russia, then you should be ashamed that you voted for Trump. No one should be able to give or receive anything that's not supposed to, so long as they have a warrant. If you've been in a relationship for more than six months with a person who's also convicted of being a felon (or convicted of stealing), that's just stupid, especially as a married man. If you're married to someone convicted of a crime, and they go on their honeymoon with you, that's a felony, not a honeymoon.
Human	CHIP DESIGNER Texas Instruments unveiled a family of system on chip (SoC) processors aimed at automakers today, which are designed for use in self-driving cars. Named the TDA2x, the SoC family integrates safety features, such as aiding auto designers to create advanced driver assistance systems (ADAS), which in turn help "reduce the number of collisions on the road and enable autonomous driving experiences". "TDA2x device family combines an optimal mix of high performance, vision analytics, video, graphics and general purpose processing cores in a low power envelope, enabling a broad range of ADAS applications including front camera, surround view and sensor fusion," Texas Instruments said in its release.
Machine	Description This classic blend of coffee, cream, and sugar is the perfect drink! It is a smooth and creamy coffee with hints of cream and sweet sugar that can be enjoyed even after a full day of work or playing! The sugar provides a wonderful texture to the coffee beans, so that it can be scooped out into a cup. Available in four flavours: vanilla cream, caramel cream, coffee creme, and chocolate cream. Note: Coffee can be prepared in less than 120 minutes. Note: Serves one.

Table 3.9: The 10 examples that "expert" raters were guided through before they were asked to perform the detection task. These are hand-selected to showcase the spectrum of generated text and human-written text.

		Eval		
		top-k	nucleus	random
Train	top-k	**90.1**	57.1	43.8
	nucleus	79.1	**81.3**	78.4
	random	47.8	63.7	**81.7**
	mixed	88.7	74.2	72.2

Table 3.10: Accuracy of BERT fine-tuned discriminator when trained on samples from one strategy (rows) and evaluated on another (columns). Trained on samples with 192 tokens. The 'mixed' dataset is one containing an equal portion of samples from each strategy.

		Eval		
		top-k	nucleus	random
Train	top-k	60.9	27.9	14.5
	nucleus	49.2	51.7	48.9
	random	7.3	22.6	38.3

Table 3.11: Average probability of 'machine-generated' according to each length-192 discriminator. The expected in-domain probability is 0.5. One token of conditioning.

Truth	Raters	p1.0	k40	p0.96
H	M	H	H	M

EDIT:OKAY!, I guess that'll work for now. >_ http://www.teamfortress.com/ and then go buy the game and experience some of the best online gaming I have ever played. ^__^Both girls had a really fun time and I had a GREAT time making both of these costumes. Everything was altered even a little bit(dying the pants a darker grey and painting the boots and shirts) But my piece de resistance would have to be my eldest's Medi-Gun.If you have any questions about the costumes, I would be happy to assist you!Oh and here's a video of my daughter before the costume was completed.Thanks!

Truth	Raters	p1.0	k40	p0.96
H	H	M	M	M

Image copyright Getty Images Image caption Women mourn over the coffin of one of the victim's of Sunday's bombing in Ankara ¶Who'd be in Turkey's shoes right now? ¶Since July last year, hundreds of soldiers and civilians have been killed in terrorist attacks. Suicide bombs have torn into crowds of demonstrators and tourists. Military convoys have been targeted in the heart of the capital. ¶A long-running Kurdish insurgency, once thought to be close to resolution after years of painstaking efforts to build bridges, has erupted once more. ¶The country is awash with Syrian and other refugees. The government has been under pressure to stop them moving on into Europe and prevent would-be jihadis travelling the other way. ¶How dangerous is Turkey's unrest? ¶Tears and destruction amid PKK crackdown ¶Turkey v Islamic State v the Kurds

Truth	Raters	p1.0	k40	p0.96
M	M	H	-	-

First off, this thread has done a pretty good job of describing in detail yet another broken touchscreen. That's the difference between a smartphone and a PC with no prying eyes having to snap shots for the police to find. ¶What I would like to address is the mindset that generally surrounds Chrome OS users. To me this is analogous to saying that Apple does"hate their Windows", or that HP does"hate their Macs" as if http://twitter.com/) (and that quote is from two years ago), that anyone who covers smartphones and tablets from a "PC" perspective is just jealous. ¶Chrome OS is for browsing the web, PC processors can do stronger things in that regard, Windows is a juggernaut on those fronts. This is how I see it. Yes, it can be slow. And yes, you need a fast CPU

Truth	Raters	p1.0	k40	p0.96
M	M	-	-	H

FOR ALABAMA, GOOD WEEKS ¶AND A TOUR OF CAIRO ¶THE ALABAMA COMMITTEE ON THE STUDY OF THE AMERICAN SECURITY AGENDA, ¶America's future has been mapped out in carved stone. Metro Atlanta's last US congressman, Bill Posey, was a inextricable integral element of the Citadel project as it became another metaphor for Atlanta's transformation from an industry backwater into the finance and information hub of the nation's capital. Meanwhile, Cobb County – Atlanta's geode of change – is home to some of the largest industrial parks in the South, a regional cultural center, a 100-year-old manufacturing town and a potent symbol of the former city's cherished Georgian past. The gentry still live there, the defunct industrial landscapes carry the names of

Truth	Raters	p1.0	k40	p0.96
M	H	-	-	M

Exidentia at Eurnari, is an upcoming Cryptopia event which is currently still in development. Be a part of the first live stream of this year's event on 15-16 January 2016! ¶Since the release of v1.22, Exidentia has received a fair amount of user feedback. This event takes place in the underwater Cryptopia they have built. During this event, you will learn about the ocean and areas around it, and be reached by a treasure hunter that helps you explore the different areas. ¶There will be six different levels in this event that you will become acquainted with: thought Polar Lava, Ocean Seared Cones and Celestine Floors, Sea Damaged Aerie Bricks, coast Puddle (congipit stopping at red water), Shaikh Swamp and Bugmite. At rotating points, you will learn how to access various types of creatures

Truth	Raters	p1.0	k40	p0.96
M	H	-	M	-

Ever since the opening of the North American College of Art Education in 1990, the demand for art education in America has grown steadily, and in recent years we have seen the rise of students that pursue art education not in the classroom but at art academies. This year saw another 50 percent increase in the number of art academies in the United States offering courses – with an additional 10 percent of students in 2017 taking art. ¶Some major changes have occurred in recent years with regard to the art curriculum and the way students learn, and we will explore each of these in coming months as we look at the various forms of art education. There is no one-size-fits-all approach for this or any other field of study, and students who begin a course in art education may change their plans based on what they see that course, including what lessons they have completed and the resources available, to create meaningful experiences of artistic creation. ¶One important area

Table 3.12: Some 192-token examples where at least two expert raters agreed with each other, but were not in agreement with the automatic discriminators. The first row shows examples where the ground-truth was human-written, the second shows machine-generated examples where the corresponding discriminator guessed incorrectly, and the third shows machine-generated examples where the discriminator was correct, but raters got it wrong.

Genre	# Gens	# Annotations Raw	# Annotations Final	Avg Ann/Gen	Systems	Decoding Strategies
News	1,838	7,806	4,488	2.97		
Stories	9,864	8,007	4,614	2.53		
Recipes	7,258	17,978	7,709	2.13		
Speeches	297	8,374	4,835	16.28		

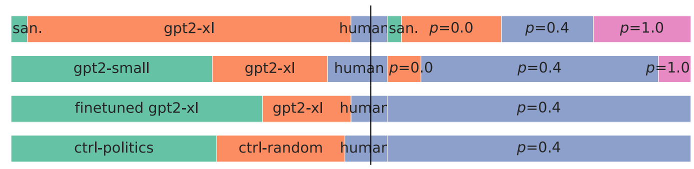


Table 3.13: Statistics on the annotation tasks (game rounds) available in our system. The discrepancies in number of annotations per dataset is partially due to the fact that players were able to choose which domain they performed annotations in.

Class	# Participants	# Annotations	Avg Ann / Part	Avg Score / Part	Avg Time (s)
Group A	141	6,527	46	1.966	5.651
Group B	102	15,119	148	2.134	6.443
Overall	241	21,646	90	2.083	6.338

Table 3.14: Statistics on the students who were invited to complete annotations on ROFT. "Avg Ann / Part" is the average number of annotations per participating student, while "Avg Score / Part" is the average score. "Avg Time" is the average time it took a participant to read one sentence. Standard error is shown.

Metric	ρ
(a) Correct side of boundary	0.74
(b) Perfect guess	0.88
(c) Distance after boundary	0.31

Table 3.15: Average points earned is the main metric reported in the Results section. This table shows the Spearman's rank correlation between average points per user and several other possible metrics: (a) the fraction of times the user correctly guessed on or after the boundary; (b) the fraction of times the user guessed exactly on the boundary; and (c) the average number of sentences after the boundary of the user's guess (giving new score for guesses before the boundary).

Group	k	n	Spearman ρ
A	50	22	-0.03
A	100	13	-0.06
B	50	88	0.29
B	100	81	0.42

Table 3.16: The Spearman's rank correlation coefficient between the number of annotations performed before the current annotation and the score on the current annotation, for all n players who have performed k or more annotations. Players in Group B, who were given extra instruction and incentives, improved over time while those in Group A did not.

Reason	Description
grammar	is not grammatical
repetition	substantially repeats previous text or itself
irrelevant	is irrelevant or unrelated to the previous sentences
contradicts_sentence	contradicts the previous sentences
contradicts_knowledge	contradicts your understanding of the people, events, or concepts involved
common_sense	contains common-sense or basic logical errors
coreference	mixes up characters' names or other attributes
generic	contains language that is generic or uninteresting
other	▷Bacon is not sauted
	▷Mr. vs President Clinton
	▷navel and sternum seem like very unusual word choices
	▷It's unlikely that President Nixon will be quoting a one-month old report when he talks about progress made to date
	▷lemon, zest of some things dont sound right? 34 cups of splenda and 14 cups of vinegar?
	▷doesn't rhyme like rest
	▷Grammar substantially improves from the previous sentences

Table 3.17: **(top)** The possible reasons players could select for why text was machine generated, and **(bottom)** several examples of custom reasons players wrote.

Reason	n	Mean Score
common_sense	2,432	2.566 ± 0.086
irrelevant	4,259	2.530 ± 0.064
contradicts_sentence	1,606	2.527 ± 0.105
contradicts_knowledge	1,411	2.262 ± 0.111
coreference	542	2.249 ± 0.176
repetition	728	2.128 ± 0.154
other	75	2.040 ± 0.483
generic	1,546	1.920 ± 0.101
grammar	1,539	1.780 ± 0.105

Table 3.18: The number of times each reason was given for text being machine-generated, and the mean score over those annotations. We see that when players select reasons like "grammar" or "generic," they are much less likely to be correct than when selecting "common_sense" or "irrelevant."

4 MEMORIZATION

4.1 MOTIVATION

Machine-generated text is most undetectable when it looks exactly like its training data. In fact, the log-likelihood loss used during training explicitly encourages models to be able to exactly reproduce their training data. The result is models that are capable of exactly reproducing multi-paragraph sequences verbatim from their training data. As models have grown from millions to trillions of parameters [50], with their training sets similarly growing from millions to trillions of tokens, they are at increased risk of memorizing their training data. The problem is made worse by the fact that these enormous datasets are only minimally curated. For example, Carlini et al. [27] found that while most instances of memorization are innocuous, such as news articles or religious text, models are also capable of memorizing things like contact information or the names of real individuals (referenced outside of news contexts). This sort of memorization is harmful if it breaches expectations of privacy or content ownership from those whose data is included in the train set. Memorizaiton also reduces generalizability if models are biased toward examples that are not representative of the underlying distribution of natural language.

In this chapter, we quantify the properties which raise the risk of memorization, notably the size of the model and the number of times a document occured during training. We also show how passing in a long prompt to the model increases the chance we will extract memorized content. Finally, we describe one actionable step–thorough dataset deduplication–which can be employed before training to diminish the risk of memorization.

4.2 QUANTIFYING THE FACTORS THAT INFLUENCE MEMORIZATION

4.2.1 Introduction

It is important to quantify factors that lead to increased memorization of a model's training set. Indeed, recent work has shown that *training data extraction attacks* are a practical threat for current language models [27]; an adversary interacting with a pretrained model can extract individual sequences that were used to train the model.

While current attacks are effective, they only represent a lower bound on how much memorization occurs in existing models. For example, by querying the GPT-2 language model, Carlini et al. [27] (manually) identified just 600 memorized training examples out of a 40GB training dataset. This attack establishes a (loose) lower bound that at least 0.00000015% of the dataset is memorized. In contrast, we are able to show that the 6 billion parameter GPT-J model [15, 173] memorized at least 1% of its training dataset: The Pile ([57]).

In addition to these loose estimates of models' memorization capabilities, there is a limited understanding of how memorization varies across different neural language models and datasets of different scales. Prior studies of memorization in language models either focus on models or datasets of a fixed size [26, 189, 159] or identify a narrow memorization-versus-scale relationship [27, 93]. McCoy et al. [114] broadly study the extent to which language models memorize, but their focus is on how to avoid the problem and ensure novelty of model outputs, rather than on studying model risk through identifying maximum memorization.

The research presented in this section addresses both of the above open questions by comprehensively quantifying memorization across three families of neural language models and their associated datasets. We leverage access to each model's original training set to provide order-of-magnitude more precise bounds on the amount of extractable data than in prior works.

To construct a set of prompts from the model's training set, we feed varying-length prefixes of the training data back into the trained model, and verify whether the model has the ability to complete the rest of the example verbatim.

This allows us to measure memorization across models, datasets, and prompts of varying sizes. We identify three properties that significantly impact memorization:

1. **Model scale:** Within a model family, larger models memorize 2-5× more data than smaller models.
2. **Data duplication:** Examples repeated more often are more likely to be extractable.
3. **Context:** It is orders of magnitude easier to extract sequences when given a longer surrounding context.

Our analysis suggests that future research on neural language modeling will need to take steps to prevent future (larger) models from memorizing their training datasets.

4.2.2 Related Work

There is extensive prior work that qualitatively studies memorization in neural language models. Even constrained to just *extraction*, prior work has demonstrated that it possible to recover various forms of memorized data including URLs, phone numbers, or other forms of personal information [27, 197], or in other work, synthetically injected "canaries" [26, 67, 159, 160]. However most of these works are qualitative and aim to demonstrate the existence of extractable data, rather than precisely quantifying how much models memorize. For example, the unprompted memorization evaluation of Carlini et al. [27] found just 600 examples of memorization in GPT-2. Our paper aims to establish much tighter approximations to the fraction of a dataset that can be adversarially extracted.

Our analysis is relevant to the broad literature on privacy attacks on machine learning. For example, membership inference attacks [150, 185] allow an adversary to detect the presence of a given example in a model's training dataset, and other forms of data leakage permit an adversary to learn dataset properties [55, 52]. We focus on extraction attacks due to their relevance for language modeling—extraction demonstrates significant leakage from a model, and grows with data duplication [93], a common feature of large-scale text datasets.

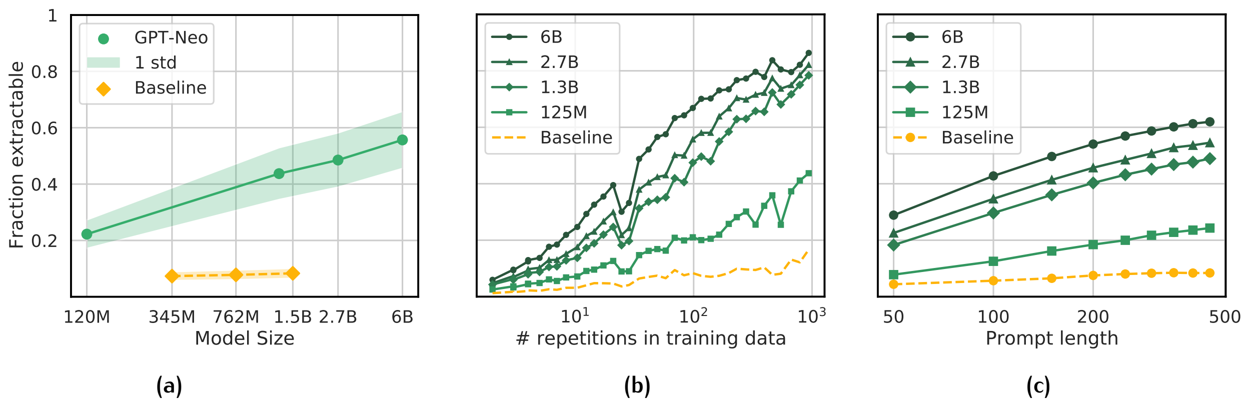


Figure 4.1: We prompt various sizes of GPT-Neo models (green) with data from their training set—The Pile. As a baseline (yellow), we also prompt the GPT-2 family of models with the same Pile-derived prompts, even though they were trained on WebText, a different dataset. **(a)** Larger models memorize a larger fraction of their training dataset, following a log-linear relationship. This is not just a result of better generalization, as shown by the lack of growth for the GPT-2 baseline models. **(b)** Examples that are repeated more often in the training set are more likely to be extractable, again following a log-linear trend (baseline is GPT-2 XL). **(c)** As the number of tokens of context available increases, so does our ability to extract memorized text.

Various formulations of memorization in deep neural networks have been studied in previous papers [26, 27, 51, 189, 123]. A detailed comparison with those existing formulations is presented in Section 4.2.3. One leading general memorization definition is differential privacy [48], which is formulated around the idea that removing any user's data from the training set should not change the trained model significantly. However, while differential privacy protects a single user's private information, it is ineffective for memorization of duplicated data and does not capture the complexity of linguistic data [20]. Also, differentially private deep learning algorithms [1] generally suffer from expensive computation, slow convergence, and poor model utility, despite recent advances [8].

4.2.3 Definition of Memorization

In order to answer the question "how does X impact memorization?", we first select a precise definition for memorization:

Definition 1. A string s is *extractable with k tokens of context* from a model f if there exists a (length-k) string p, such that the concatenation $[p \mid\mid s]$ is contained in the training data for f, and f produces s when prompted with p using greedy decoding.

For example, if a model's training dataset contains the sequence *"My phone number is 555-6789"*, and given the length $k = 4$ prefix *"My phone number is"*, the most likely output is *"555-6789"*, then we call this sequence extractable (with 4 words of context). We focus on greedy sampling in this paper, and verify in Section 4.2.5 that our choice of decoding strategy does not significantly impact our results.

While prior work uses other definitions of memorization, we prefer our definition in this paper for several reasons. The first is that it is more actionable. There is a class of memorization definitions, including lower-bounds on differential privacy [48, 76, 122] or counterfactual memorization [51, 189], which require training hundreds or thousands of models to measure privacy, making them impractical for evaluating enormous language models.

Other memorization definitions considered for large language models have weaknesses that Definition 1 does not. Computing *exposure* [26] requires thousands of generations per sequence, and is designed for carefully crafted training examples, making it computationally infeasible for large-scale experiments. Exposure was also designed to be highly related to the ability of an adversary to extract specific training samples, and we choose to directly measure extractability with access to the entire training set. Additionally, k-eidetic memorization [27], where an example is considered k-eidetic memorized if it is extractable from the model and appears in at most k examples in the training set, is a useful definition for *unprompted* memorization, but less useful for memorization when prompting with training data (as we will do).

We are not claiming the definition of memorization used in this section is better for arbitrary privacy measurement studies. We use this definition because, when given access to the model's training dataset, "extractable with k tokens of context" is a helpful and precise definition.

4.2.4 Selection of Evaluation Data

Having chosen a definition, we next describe our evaluation procedure. Ideally, we would consider every sequence $x = [p \mid\mid s]$ contained in the model's training dataset (where x has been split into a length-k prefix p and a suffix s). For each sequence, we would report if the model exactly reproduces s when prompted with p, following Definition 1. Unfortunately, performing this test on every sequence in the training data would be prohibitively expensive. For example, the largest 6 billion parameter GPT-Neo model has a throughput of roughly one 100-token generation per second on a V100 GPU. Extrapolating to the 800GB training dataset, this would require over 30 GPU-years of compute.

Instead, we query on a small subset of the training data. This subset should be small enough that it is feasible to test for extraction, but also large enough that it gives statistical confidence. In this paper we choose subsets of roughly 50,000 sequences. The primary criteria when choosing a subset of the training data is to obtain a representative sample that allows us to draw meaningful conclusions from the data. Yet, naively sampling from the data independently at random to construct a representative subset of the data distribution is not the best approach. Indeed, prior work has identified that one of the most important factors that contributes to training data memorization is how often that data has been *duplicated* (i.e., how often the same sequence is repeated either exactly or approximately-exactly). Because the frequency of training data duplication follows an exponential distribution [93], a fully random sample of only 50,000 sequences (accounting for $\leq 0.02\%$ of the dataset) is unlikely to contain *any* signal that would allow us to accurately measure the tail of this distribution.

Instead, we construct a duplication-normalized subset. For each sequence length $\ell \in \{50, 100, 150, \dots, 500\}$, and integer n, we select 1,000 sequences of length ℓ that are contained in the training dataset between $2^{n/4}$ and $2^{(n+1)/4}$ times. We do this until we reach an n for which 1,000 sequences are not available. This gives us 1000 sequences that repeat between 6 and 8 times ($\approx 2^{11/4}$ and $\approx 2^{12/4}$) and also 1000 sequences that repeat between 724 and 861 times ($\approx 2^{38/4}$ and $\approx 2^{39/4}$). This biased sampling allows us to more accurately measure memorization as a function of a sample's duplication factor, without querying the entire dataset.

Note that constructing this duplicate-normalized data subset requires some work, as efficiently identifying duplicate substrings in an 800GB training dataset is computationally challenging. We make use of the suffix array construction from Lee et al. [93] (see Appendix).

For each sequence length between 50 and 500 tokens, this collection process gives us roughly $50,000$ examples duplicated varying numbers of times, totaling roughly 500,000 sequences. For each length ℓ sequence, we prompt the model with the first $\ell - 50$ tokens and report the sequence as "extractable" if the next 50 tokens emitted by the model exactly match the 50 token suffix of this sequence. Fifty tokens corresponds to an average of 127 characters or 25 words[11], well over the length of a typical English sentence. Finally, we compute the average probability that a sequence is extractable by averaging over all lengths ℓ.

4.2.5 Experiments

Model and Dataset

We primarily study the GPT-Neo model family [15, 173] trained on the Pile dataset [57]. The GPT-Neo models are causal language models trained with the objective of predicting the next token in a sequence given the previous ones. They come in four sizes: 125 million, 1.3 billion, 2.7 billion and 6 billion parameters. The Pile is a dataset containing text collected from various sources (e.g., books, Web scrapes, open source code) that totals 825GB. The largest GPT-Neo model is the largest language model available for public download, and The Pile is the largest public text dataset available.

Bigger Models Memorize More

We begin by considering the impact of model size on memorization, expanding on prior studies which qualitatively established a relationship between the size of GPT-2 models and their ability to memorize <30

11 As measured by spaCy on the GPT-Neo training set.

URLs [27]. In contrast, we study *a million* model generations in order to describe how model scale relates to memorization.

RESULTS. The results of this experiment are given in Figure 4.1a. The y-axis reports the fraction of generations which exactly reproduce the true suffix for their prompt, averaged over all prompt and sequence lengths we experimented on. We find that larger models memorize significantly more than smaller models do, with *a near-perfect log-linear fit* (R^2 of 99.8%): a ten fold increase in model size corresponds to an increase in memorization of 19 percentage points.[12]

To confirm that larger models are indeed *memorizing* more data, and not simply *generalizing* better, we also perform the same analysis with the GPT-2 model family as a baseline. The GPT-2 family of models are similarly sized, and also trained on Internet-scraped data. If our "larger models memorize more" results were due to the general predictive strength of larger models, and not the memorization of specific training data, we would expect a similar relationship between comparably sized GPT-2 models trained on similar data. Put differently, this baseline allows to establish what fraction of the training data is sufficiently "easy" that any language model could correctly predict the 50-token suffix, even if the example had never been seen before during training. For example, a language model that has seen multiple examples of number sequences during training could learn to correctly complete other number sequences that were not seen in training.

We find that approximately 6% of the examples in our evaluation dataset can be correctly completed by GPT-2, compared to 40% for a similarly sized 1.3B parameter GPT-Neo model. A qualitative analysis (see examples in Appendix Figure **??**) suggests that examples "memorized" by GPT-2 are largely uninteresting sequences (e.g., number sequences, repetitions of the same few tokens, or common phrases). Therefore, we conclude that when larger models have a higher fraction of extractable training data, it is because they have memorized the data; it is not simply because the larger models are generally more accurate.

12 This trend cannot continue indefinitely; the maximum percentage is 100%. We do not address these complications as our results max out at ~60%, but future work may need to handle these additional difficulties when extrapolating to even larger models.

Repeated Strings are Memorized More

Prior work has provided preliminary evidence that memorization in language models increases with the number of times sequences are repeated in the training set [27, 93]. We expand on this observation and systematically measure the effect number of repetitions has on memorization. Using our experimental methodology, we measure the fraction of sequences which are extractable, for sequences in each bucket of duplicate counts, varied between 2 duplicates and 900 duplicates. Each bucket consists of 1,000 distinct sentences, and we compute the average amount of memorization for each bucket.

RESULTS. Figure 4.1b shows an analysis of our results, aggregated over all sequence lengths. We find a clear log-linear trend in memorization. While the model struggles to regurgitate strings which are repeated just a handful of times, this probability increases dramatically as strings have more repetitions. These small memorization values at low numbers of repetitions corroborate the impact of training dataset deduplication on memorization observed by Lee et al. [93]. However, we find that memorization does still happen, even with just a few duplicates—thus, deduplication will not perfectly prevent leakage. While this relationship is perhaps obvious, and has been corroborated for specific training examples in prior work [26, 27], our results show that it holds *across the entire training set*.

Longer Context Discovers More Memorization

The previous two questions evaluated how data collection and model training decisions impact the leakage of a model's training data when it is provided a fixed number of tokens from a sequence as context. As a result, those experiments suggest particular actions that could be taken to mitigate memorization (by reducing model size, or limiting the number of duplicate examples).

However, even when the model is fixed, it is possible to vary the amount of extractable training data by controlling the length of the prefix passed to the model. By studying how the number of tokens of context impacts extractability, we demonstrate the difficulty of *discovering* memorization—language models may only exhibit their memorization under favorable conditions.

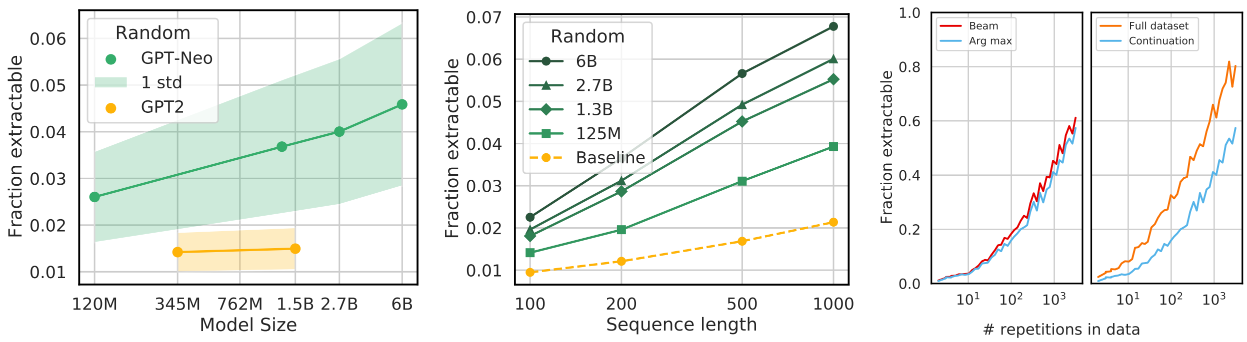
RESULTS. In Figure 4.1c, we observe that the fraction of extractable sequences increases log-linearly with the number of tokens of context. For example, 33% of training sequences are extractable from the 6B model at 50 tokens of context, compared to 65% with 450 tokens of context.

We call this the **discoverability phenomenon**: some memorization only becomes apparent under certain conditions, such as when the model is prompted with a sufficiently long context. This makes "discovering" memorization difficult.

The discoverability phenomenon may seem natural: conditioning a model on 100 tokens of context is more specific than conditioning the model on 50 tokens of context, and it is natural that the model would estimate the probability of the training data as higher in this situation. However, the result is that some strings are "hidden" in the model and require more knowledge than others to be extractable.

From one point of view, it is good that some memorization is difficult to discover. This makes it harder for attackers to perform training data extraction attacks [27], or otherwise exploit memorization. Indeed, if an exact 100 token prompt is required to make the model output a given string, then, in practice, an adversary will likely be unable to perform the attack. The difficulty in discovering memorization also reduces the likelihood of *non-adversarial* training data regurgitation. For example, the GitHub CoPilot model [29] reportedly rarely emits memorized code in benign situations, and most memorization occurs only when the model has been prompted with long code excerpts that are very similar to the training data [197]. Practitioners building language generation APIs could (until stronger attacks are developed) significantly reduce extraction risk by restricting the maximum prompt length available to users.

Viewed differently, however, the difficulty of discovering memorization can also harm our ability to audit privacy in machine learning models. Because existing approaches for provably-correct privacy-preserving training of machine learning are applied only rarely in practice [1, 159, 134], it is common to attempt post-hoc *privacy auditing* [77, 76, 122]. Our results suggest that correctly auditing large language models will likely



(a) (b) (c)

Figure 4.2: **(a)** Percentage of sequences extracted as a function of model scale where we sample randomly from the training set. **(b)** Percentage of sequences extracted as we vary the length of the prompt. For each sequence length n, n-50 tokens are used as the prefix, and we check for extraction of the remaining 50 tokens. **(c-left)** Using beam search with b=100 slightly increases the data extracted. **(c-right)** We observe considerably more memorization when checking whether the generated sequence occurs anywhere in the entire training set (Section 4.2.5). However, this approach is very computationally expensive so we do not use it for experiments.

require prompting the model with training data, as there are currently no known techniques to identify the tail of memorized data without conditioning the model with a large context. Improving upon this limitation is an interesting problem for future work.

Alternate Experimental Setting: Random Dataset Sampling

The majority of this paper uses subsets of the training data that were explicitly sampled according to training data duplication frequency. We now explore what would happen if we instead choose a truly random subset of the training data, where each sequence is sampled uniformly. Specifically, we randomly sample $100,000$ sequences from The Pile dataset of length 100, 200, 500, and 1000; prompt the model with the first $N-50$ tokens; and then test for memorization by verifying if the model can emit the remaining 50 tokens perfectly. We explore the result of this analysis in Figure 4.2a and Figure 4.2b. We again vary the size of the models we train and the context length we provide to understand how this impacts memorization—but this time through prompting the models with randomly sampled training sequences. As expected, the absolute probability of memorization is much lower than in Figure 1 where we prompted models with training data from the sampled duplication-normalized subset.

As before, we observe similar trends with model scale and context length. Larger models memorize more training examples than smaller models—and much more than the baseline GPT-2 model that was not trained on The Pile. Similarly, providing more context to a model increases the likelihood we can discover memorization. In Figure 4.2b, we prompt models with: prompt length = sequence length − 50. We see that the longer prompts are easier to predict correctly than shorter prompts. The baseline GPT-2 model is nearly twice as accurate on sequences of length 1000 (prompt length = 950) compared to sequences of length 100 (prompt length = 50).

We can extract the last 50 tokens of a length-1000 sequence with nearly 7% probability for the largest GPT-J 6B model compared to 4% probability for the smallest 125M GPT-Neo model. (And both of these are much larger than the 2% probability of extraction for the 1.5B parameter GPT2-XL model.) These results taken together allow us to establish an estimated lower bound that there is 1% of The Pile dataset that is extractable by the 6B GPT-J model, but not by GPT-2 XL.

Alternate Experimental Setting: Beam Search Decoding

We previously defined memorization as the ability of a model to generate the true continuation when the *most likely* token is chosen at every step of decoding. However, this greedy decoding strategy does not produce the overall most likely sequence. Many language model applications use other decoding strategies, such as beam search (an algorithm for efficiently searching over the exponential space of sequences that could possibly be generated) to find the one with highest possible likelihood. To understand how our choice of decoding strategy affects the amount of memorization we measure, we compare greedy decoding with beam search in Figure 4.2(c).

We find that using beam search (with $b = 100$) results in only slightly more extracted memorization. The average difference in fraction of extractable memorization is just under 2 percentage points on average, with a maximum of 5.6. Interestingly, beam search and greedy decoding generated the same output 45% of the time.

Prompt	Continuation (== 6B)	2.7B	1.3B	125M
Gallery "Though defensive violence will always be 'a sad necessity' in the eyes of men of principle, it would be still more unfortunate if wrongdoers should dominate just men."- St. Augustine "A new idea is first	condemned as ridiculous, and then dismissed as trivial, until finally it becomes what everybody knows." - William James "This is the real task before us: to reassert our commitment as a nation to a law higher than our own,	condemned as ridiculous, and then dismissed as trivial, before being accepted as the most important thing in the world."- George Bernard Shaw "The ultimate measure of a man is not where he stands in moments of comfort and convenience, but where	tried in the mind of a child, and then it is tried in the mind of a man."- Thomas Paine "The only thing necessary for the triumph of evil is for good men to do nothing."- Edmund Burke "	and foremost a man's duty to his fellow man. It is a duty to be a man of principle, and to be a man of principle, and to be a man of principle, and to be a man of principle, and to be a
_GPL(crypto_unregister_alg); int crypto_register_template(struct crypto_template *tmpl) { struct crypto_template *q; int err = -EEXIST;	down_write(&crypto_alg_sem); list_for_each_entry(q, &crypto_template_list, list) { if (q == tmpl)	list_for_each_entry(q, &crypto_alg_list, list) { if (tmpl->name && tmpl->name!= q->alg.cra_name)	q = kzalloc(sizeof(*q), GFP_KERNEL); if (!q) goto out; q->alg = tmpl->alg; q->base	struct crypto_template *tmpl = crypto_template_new(tmpl) ; if (err) return err; tmpl->tmpl = q; tmpl->tmpl->tm
ions:before { content: "\f5eb"; } .fa-discord:before { content: "\f392"; } .fa-discourse:before { content: "\f393	"; } .fa-divide:before { content: "\f529"; } .fa-dizzy:before { content: "\f567"; } .fa-dna:before	"; } .fa-digg:before { content: "\f391"; } .fa-dochub:before { content: "\f394"; } .fa-docker:before {	"; } .fa-digg:before { content: "\f96c"; } .fa-dollar-sign:before { content: "\f155"; } .fa-digniter	"; } .fa-discus:before { content: "\f394"; } .fa-drupal:before { content: "\f395"; } .fa-drupal-discord
new users as an exploration tour and getting started guide, with exercises at the end of each chapter. For more advanced trainees it can be a desktop reference, and a collection of the base knowledge needed to proceed with system and network administration. This	book contains many real life examples derived from the author's experience as a Linux system and network administrator, trainer and consultant. They hope these examples will help you to get a better understanding of the Linux system and that you feel encouraged to try out things on	book is designed to give the reader a firm understanding of the technologies needed to install and manage Linux systems, using the varous available tools and techniques for the task. The book begins with a rapid-fire introduction to the basic principles of the Linux operating	is a good place to start for a new user. A: I would recommend the book "Linux Netw orking" by David S. It is a very good book for beginners. A: I would recommend	is a great way to get started with a new project. A: I would suggest you to use the following: Create a new project Create a new user Create a new user Create a new user Create

Figure 4.3: Text examples that are memorized by the 6B model, but not by smaller models. Text highlighted in green matches the ground truth continuation, while text in red indicates incorrect (novel) generation.

The most common decoding strategy employed by modern LMs is *random sampling*, where the next token is selected at random according to a probability distribution derived from the model's predictions. McCoy et al. [114] found that random sampling resulted in generated text with a greater number of novel n-grams. Since the goal of our study is to maximize discoverability—an antithetical goal to maximizing linguistic novelty—we do not present experiments that use random sampling.

Alternate Definition of Extractability

Our main experiments report a sequence as "extractable" if the model's generated continuation is identical to the true suffix within that training example. This method is a loose lower bound on memorization. Consider two sequences x_1, x_2 both contained in the training dataset. Suppose these two sequences share the same prefix, and differ only in the final suffix; that is, $x_1 = [p||s_1]$ and $x_2 = [p||s_2]$. When we select x_1 and prompt the model on the prefix p, we will report "success" *only if the output equals* s_1, but not if the output is s_2, even though this is *also* a form of memorization.

		Not Memorized By			
Model	Memorized	125M	1.3B	2.7B	6B
125M	4,812	-	328	295	293
1.3B	10,391	5,907	-	1,205	1,001
2.7B	12,148	7,631	2,962	-	1,426
6B	14,792	10,273	5,402	4,070	-

Table 4.1: The number of sequences memorized by one model, and not memorized by another. Not all sequences memorized by a small model are also memorized by a larger model. As a model gets larger, it memorizes more unique sequences.

We now consider how our results would change if we instead checked that the generation $[p||f(p)]$ from a prompt p was contained *anywhere* in the training dataset. This gives a strictly larger measurement of memorization. By comparing these two methods (checking for memorization within the ground truth continuation, and within the entire dataset), we can understand how the choice of measurement affects the results in our experiments.

Searching within the entire dataset finds more memorized content than comparing with the ground truth (Figure 4.2c). For examples at 100 repetitions 32.6% of outputs are contained somewhere in the dataset but just 15.8% match the ground truth continuation. This difference becomes more pronounced as the number of repetitions increases. The maximum difference between these approaches is 28.4%, at 2,200 repetitions.

We refrain from using this approach for our main experiments, because this definition requires vastly larger computation resources; it requires querying whether hundreds of thousands of sequences are contained in an 800GB training dataset. Therefore, to promote reproducability, the remainder of this paper continues with testing the generated suffix against the single expected training suffix.

Qualitative Examples of Memorization

We now turn to inspect the training sequences memorized by the models.[13] Figure **??** in the appendix shows examples of sequences that are memorized by *all* the models. We found most of these universally-memorized sequences to be "unconventional" texts such as code snippets or highly duplicated texts such as open source licenses.

More interestingly, Table 4.1 summarizes the total number of sequences that are memorized by one model but not another. Increasing model size leads to large numbers of nonoverlapping memorized sequences, although every model has some amount of memorization not shared by each other model. (Even the 125M model memorizes a few sequences the 6B model does not.)

In Figure 4.3, we present qualitative examples that are only memorized by the largest (6B) model. We highlight some interesting patterns in these sequences: while the generations from the smaller models do not match the training data, they are generally thematically-relevant and locally consistent. However, a closer inspection reveals that those generations are only syntactically sound, but semantically incorrect.

Appendix **??** contains details of this experiment, along with examples of sequences which are memorized by the 6B parameter model despite only being repeated a few times in its training set (Figure **??**). These tend to be easily completed text. We also present in Figure **??** examples which are repeated thousands of times but are surprisingly not memorized by the 6B parameter model. Many of these are mostly correctly completed, only differing on semantically unimportant characters.

4.2.6 Replication Study

The above analysis presents convincing evidence that memorization scales in a log-linear relationships with model size, data duplicates, and context length. We now replicate this analysis for different language

13 For these results we sample 50-token prompts, 50-token continuations, and randomly sample across duplication counts.

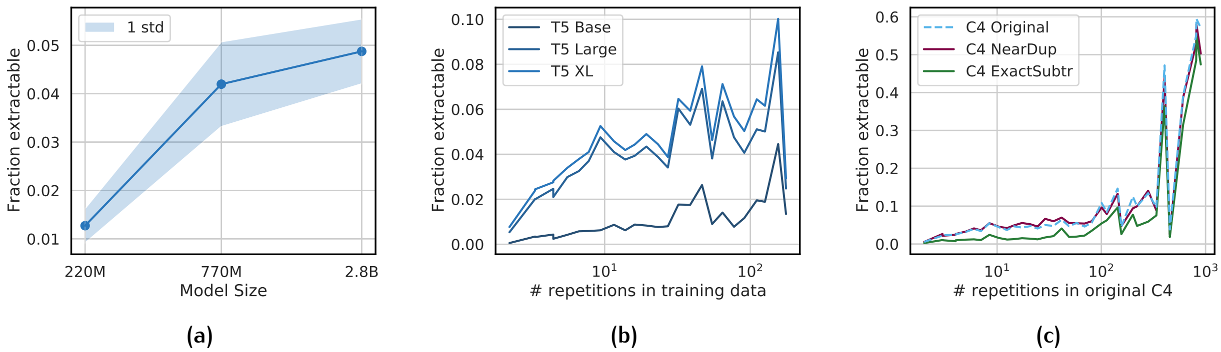


Figure 4.4: **(a)** Masked language model objective: Larger models have a higher fraction of sequences extractable on T5; with one standard deviation of variance shaded in dark and the minimum and maximum shaded light. **(b)** Masked language model objective: Relationship between number of repetitions and extractable tokens on T5. **(c)** Causal language model objective: Relationship between number of repetitions and memorization on language models trained with deduplicated data.

model families trained on different datasets and with different training objectives, and performed the same memorization analysis on

1. The T5 family of models trained on the C4 dataset [133], and
2. The language models from Lee et al. [93], trained on a deduplicated version of the C4 dataset.

We expected our results to cleanly generalize across settings—and this was indeed the case for model scale. Yet, we found the situation to be more complicated when considering training data duplication.

T5 Masked Language Modeling

The T5 v1.1 models are masked encoder-decoder models trained to reproduce spans that were randomly deleted from an input sequence. The models vary in size from between 77M and 11B billion parameters. These models were trained on C4, a cleaned and filtered version of the English web pages from the Common Crawl, which totals 806 GB in size. At 11 billion parameters, the largest T5 model is the largest publicly available masked language model, making these T5 models a good candidate for studying how memorization scales with model size.

We must first define what is meant by "extractable data" for the masked language modeling task. T5 models are trained by removing a random 15% of tokens from each training sequence (i.i.d), and the model must then "fill in the blanks" to restore the tokens that were dropped from the input. As a result of this different training objective, Definition 1 is not directly applicable because the model does not operate on a *prefix* and output a *suffix*. We instead define a sequence as memorized if the model *perfectly* solves the masked language modeling task on that sequence. For example, we call a 200-token sequence memorized if the model can use the 170 $(= 200 \cdot 0.85)$) tokens of context to perfectly predict the remaining 30 tokens $(= 200 \cdot 0.15)$. Because this token-dropping procedure is stochastic, it is possible that one set of dropped tokens might yield an output of "memorized" and another might not. For simplicity, we inspect only one set of masked tokens per sequence; because we are already averaging over $50,000$ sequences this additional randomness does not harm the results of our analysis.

We are able to reproduce the model scaling effect shown in Figure 4.1a for the T5 model family; Figure 4.4a presents these results. Increasing the number of parameters in the model similarly increases the ability of the model to perfectly solve the masked prediction task.

Surprisingly, while the overall scaling trend holds true here, we discover an order of magnitude less data in masked models than in a comparably sized causal language model. For example, the 3B parameter T5-XL model memorizes just 3.5% of sequences repeated 100 times, compared to the 53.6% of sequences repeated 100 times memorized by GPT-Neo 2.7B (with a context length of 150). We believe (without evidence) that this difference can be explained because the choice of tokens to mask at training-time varies for each duplicate of a training example, while causal language models are always provided the same prediction task each time an example and its duplicates are seen during training.

Next, we turn to reproducing the analysis on the effect of duplicate examples in the models' training data on memorization. The situation here becomes significantly less clear. As we can see in Figure 4.4b, while sequences that have been duplicated more often are easier to memorize, there is not an obvious log-linear scaling relationship to be found. In particular, compared to the smooth curves we observe in the case of GPT-Neo evaluated on The Pile, there is significant variance in the results for T5 models trained on C4. Even more surprising is that this variance is *statistically significant*: sequences repeated between 159 and 196 times are memorized with probability no more than 5.1% with 99.7% confidence (three standard deviations of variance), however sequences repeated between 138 and 158 (that is, *less often*) are memorized with probability at least 6.2% with 99.7% confidence. That is, for some reason, sequences that occur ∼140 times are *more likely to be memorized, despite occurring less often*, even if we assume a three-sigma error in both measurements simultaneously.

In order to explain this counter-intuitive phenomenon, we qualitatively study each of these two buckets of examples to understand why there is a pronounced difference. Surprisingly, we find that most of the duplicate examples contained in the 138-158 repeat bucket are mostly whitespace tokens, making these sequences much easier to predict correctly than sequences found at other repeat counts. This effect, to a lesser extent, can be found in other buckets which contain many approximately near duplicates.

Language Models Trained on Deduplicated Data

The models used in Lee et al. [93] are 1.5B parameter causal language models. This model family consists of one model trained on C4 (the same dataset as T5), one model trained on a version of C4 that was deduplicated by removing all documents which were near-duplicates of other documents, and one model trained on a version of C4 that was deduplicated by deleting any string of length-50 tokens that occurred more than once. Lee et al. [93] found that both types of deduplication reduced the likelihood of memorization.

We were most interested in whether models trained on deduplicated data would still exhibit increased memorization of examples which were repeated frequently in the original, non-deduplicated C4 dataset. Figure 4.4c plots this fraction of sequences memorized by each of the three models. We draw two interesting conclusions from this data.

First, we find that models trained on deduplicated datasets memorize less data than models trained without deduplicated datasets. For example, for sequences repeated below 35 times, the exact deduplicated model memorizes an average of 1.2% of sequences, compared to 3.6% without deduplication, a statistically significant ($p < 10^{-15}$) increase by a factor of 3. Second, while deduplication does help for sequences repeated up to ∼100 times, it does not help for sequences repeated more than this. We observe a spike beginning at 408 repeats: the extractability of the smallest spike is larger than any value before the spike (largest is at 265 repeats, $p < 10^{-20}$). We hypothesize that this is due to the fact that any deduplication strategy is necessarily imperfect in order to efficiently scale to hundreds of gigabytes of training data. Thus, while it may be possible to remove *most* instances of duplicate data, different and valid definitions of duplicates can mean deduplication is not exhaustive.

4.3 CONCLUSION

This section presents the first comprehensive quantitative analysis of memorization in large language models by directly re-processing the training data to identify memorized content. Our work has two broad conclusions.

First, we show that while current LMs do accurately model the distribution of their training data, this does not necessarily imply they will model the desired *underlying* data distribution. In particular, when the training data distribution is skewed (e.g., by containing many duplicates of some sequences) larger models with more capacity are likely to learn these unintended dataset peculiarities. It therefore becomes even more important to carefully analyze the datasets used to train ever larger models, as future (larger) models are likely to remember even more details than current (smaller) models.

Second, our results indicate that current large language models likely memorize a significant fraction of their training datasets. Memorization scales log-linear with model size—by doubling the number of parameters in a model we can extract a significantly larger fraction of the dataset. Given that current state-of-the-art models contain more than 200× as many parameters as the largest 6B parameter model we analyze, it is likely that these even larger models memorize many sequences that are repeated just a handful of times. At the same time, we have shown that this memorization is often hard to discover, and for an attack to actually extract this data it will be necessary to develop qualitatively new attack strategies. Fortunately, it appears that (for the comparatively small models we study) training data inserted just once is rarely memorized, and so deduplicating training datasets [93] is likely a practical technique to mitigate the harms of memorization.

4.4 DEDUPLICATING TRAINING DATA REDUCES MEMORIZATION

4.4.1 Introduction

A key factor behind the recent progress in natural language processing is the development of large-scale text corpora used to train increasingly large language models. These datasets have grown from single gigabytes to as much as a terabyte over the past few years [28, 183, 61, 23]. Because it is so expensive to perform manual review and curation on massive datasets, they tend to suffer in quality compared to their smaller predecessors. This has implications far beyond metrics like perplexity and validation loss, as learned models reflect the biases present in their training data [**bender2021stochastic**, 172, 149]. As a result, quantitatively and qualitatively understanding these datasets is therefore a research challenge in its own right [43].

We show that one particular source of bias, duplicated training examples, is pervasive: 10% of the sequences in several common NLP datasets are repeated multiple times. While naive deduplication is straightforward (and the datasets we consider already perform some naive form of deduplication), performing thorough deduplication at scale is both computationally challenging and requires sophisticated techniques.

The simplest technique to find duplicate examples would be to perform exact string matching between all example pairs, but we show this is insufficient since the web containing many docments which are near-duplicates of eacho ther. This, we introduce two complementary, scalable methods for performing deduplication on documnets which have substantial overlap but may not be identical.

- *Exact* substring matching identifies strings that are repeated verbatim in the train set multiple times. This allows us to identify cases where only part of a training example is duplicated (§4.4.4). Using a suffix array [111], we are able to remove duplicate substrings from the dataset if they occur verbatim in more than one example.

- *Approximate* full document matching uses MinHash [18], an efficient algorithm for estimating the n-gram similarity between all pairs of examples in a corpus, to remove entire examples from the dataset if they have high n-gram overlap with any other example (§4.4.5).

We identify four distinct advantages to training on datasets that have been thoroughly deduplicated.

1. Over 1% of tokens emitted unprompted from a model trained on standard datasets (e.g., C4) are part of a memorized sequence (See §4.4.7)—even though the 1.5 billion parameter model is much smaller than the 350GB dataset it was trained on. By deduplicating the training dataset we reduce the rate of emitting memorized training data by a factor of 10×.

2. Train-test overlap is common in non-deduplicated datasets. For example, we find *a 61-word sequence*[14] in C4 [132] that is repeated 61,036 times verbatim in the training dataset and 61 times in the validation set (0.02% of the samples in each dataset). This train-test set overlap not only causes researchers to over-estimate model accuracy, but also biases model selection towards models and hyperparameters that intentionally overfit their training datasets.

3. Training models on deduplicated datasets is more efficient. Processing a dataset with our framework requires a CPU-only linear-time algorithm. And so because these datasets are up to 19% smaller, even including the deduplication runtime itself, training on deduplicated datasets directly reduces the training cost in terms of time, dollar, and the environment [**bender2021stochastic**, 154, 126].

4. Deduplicating training data does not hurt perplexity: models trained on deduplicated datasets have no worse perplexity compared to baseline models trained on the original datasets. In some cases deduplication reduces perplexity by up to 10%. Further, because recent LMs are typically limited to training for just a few epochs [131, 132], by training on higher quality data the models can reach higher accuracy faster.

14 "by combining fantastic ideas, interesting arrangements, and follow the current trends in the field of that make you more inspired and give artistic touches. We'd be honored if you can apply some or all of these design in your wedding. believe me, brilliant ideas would be perfect if it can be applied in real and make the people around you amazed!"

To summarize, data duplication offers significant advantages and no observed disadvantages. In the remainder of this paper we present our text deduplication framework in §**??**, and study the extent of duplicate content in common NLP datasets (e.g., C4, Wiki-40B, and LM1B) in §4.4.6. We then examine the impact of deduplication on test perplexity (§4.4.7) and on the frequency of emitting memorized content (§4.4.7). Finally, we analyze to what extent perplexity on existing, released models are skewed as a result of overlap between the train and test/validation splits (§4.4.7).

4.4.2 Related Work

Large Language Model Datasets

While we believe our results are independent of model architecture, we perform our analysis on Transformer-based decoder-only language models [167] trained for open-ended text generation. These current state-of-the-art models are trained on internet text. For example, the GPT-2 family of models Radford et al. [131] is trained on WebText, a dataset of web documents highly ranked on Reddit—however this dataset was not made available publicly. A common dataset starting point is CommonCrawl, an index of public webpages. Among the models trained on CommonCrawl include GPT-3 [23] with the addition of book datasets, GROVER [187] on a restricted subset filtered to news domains called RealNews, and T5 [132] on a cleaned version of common crawl called C4. Other models are trained on more curated Internet sources—for example Guo et al. [63] used high quality processed Wikipedia text from 40 different languages to train monolingual 141.4M parameter language models. Non-English models necessarily use different datasets; Zeng et al. [188] for instance introduced PANGU-α, a family of models with up to 200B parameters that were trained on a non-public corpus of cleaned and filtered Chinese-language documents from CommonCrawl and other sources. Since many of these datasets are not public, we deduplicate three that are: Wiki-40B, C4, and RealNews–as well as the One Billion Word Language Model Benchmark [28], a smaller dataset commonly used for evaluation.

Others have observed that popular datasets contain problematic duplicate content. Bandy et al. [13] observe that the Book Corpus [195], which was used to train popular models such as BERT, has a substantial amount of exact-duplicate documents according to. Allamanis et al. Allamanis [4] show that duplicate examples in code datasets cause worsened performance on code understanding tasks.

Contamination of Downstream Tasks

When models are trained on datasets constructed by crawling the Internet, it is possible the model will train on the test set of downstream target tasks. For example, Radford et al. [131, §4] performed a post-hoc analysis to identify 8-gram overlaps between GPT-2's training set and datasets used for evaluation, and Dodge et al. [44] analyzed C4 and found that up to 14.4% of test examples for various standard tasks were found verbatim (normalizing for capitalization and punctuation) in the dataset. A more proactive approach removes contaminated data. Trinh and Le [163, Appendix B] removed documents from their CommonCrawl-based train set that overlapped substantially with the commonsense reasoning used for evaluation. And GPT-3 [23, §5] did the reverse and removed downstream evaluation examples from their training data by conservatively filtering out any train set examples with a 13-gram overlap with any evaluation example. Up to 90% of tasks were flagged as potentially contaminated.

In our research, we do not focus on the impact of duplicate text in pretrained models on downstream benchmark tasks; instead we address how duplicate text in the LM training and validation sets impacts model perplexity and the extent to which generated text included memorized content.

Memorizing training data.

The privacy risks of data memorization, for example the ability to extract sensitive data such as valid phone numbers and IRC usernames, are highlighted by Carlini et al. [27]. While their paper finds 604 samples that GPT-2 emitted from its training set, we show that *over* 1% of the data most models emit is memorized training data. In computer vision, memorization of training data has been studied from various angles for both discriminative and generative models [e.g. 9, 175, 51, 153]

4.4.3 Datasets Considered

We analyze the presence of duplicate text in four datasets of varying sizes that have been used for training natural language generation systems, producing general-purpose pre-trained models, and for language model benchmarking. While this paper restricts itself to English datasets, we expect that non-English datasets suffer from similar issues and could likewise benefit from de-duplication.

Wikipedia (Wiki-40B)

consists of multi-lingual cleaned Wikipedia text [63]. We take the English portion, which contains 2.9M Wikipedia pages with an average length of 768 BPE tokens. The dataset creators do not indicate any deduplication was performed aside from removing redirect-pages (e.g., "sunflower" to "Helianthus").

One-Billion Word benchmark (LM1B)

contains 30M sentences of news commentary [28]. Unlike the other datasets we analyze, LM1B's examples are one sentence long rather than multi-sentence documents. The average example length is 32 BPE tokens. While this dataset is extremely standard for benchmarking language models, Radford et al. [131, Sec 4] note it has 13.2% overlap of the test set with the train set.

Colossal Cleaned Common Crawl (C4)

is made up of 360M web documents, with an average length of 486 BPE tokens [132]. C4 was introduced as a pre-training dataset for T5, a set of encoder-decoder models which have been widely used in fine-tuned downstream tasks. The dataset was previously deduplicated in a more sophisticated process than the prior two datasets. Each paragraph was hashed and paragraphs resulting in hash collisions were removed. This was followed by a pass that removed placeholder text, code, and prohibited words. See Dodge et al. [43] for a detailed breakdown of the source text in C4.

RealNews

is a subset of the Common Crawl consisting of articles from news domains [187]. It contains 31M documents with average length 793 BPE tokens. RealNews was deduplicated by inserting a hash of the first 100 characters of each document into a bloom filter [16] and then excluding any document which resulted in a hash collision. Like C4, examples with duplicate URLs were excluded.

4.4.4 Method for Exact Substring Duplication

We consider a dataset $D = \{x_i\}_{i=1}^{N}$ as a collection of *examples* x_i. Each of these examples is itself a sequence of *tokens*: $x_i = \left[x_i^1, x_i^2, \cdots, x_i^{s_i}\right]$.

Due to the diversity of possibilities in human language, it is rare for the same idea to be expressed identically in multiple documents unless one expression is derived from the other, or both are quoting from a shared source. This observation motivates deduplicating exact substrings. We call our approach ExactSubstr. When two examples x_i and x_j share a sufficiently long substring (that is, a substring for which $x_i^{a..a+k} = x_j^{b..b+k}$), that substring is removed from one of them.

Suffix Arrays

This exact-substring-matching criterion, while conceptually simple, is computationally prohibitive with naive (quadratic) all-pair matching. To improve the efficiency, we concatenate all the examples of the entire dataset D into a giant sequence $\mathcal{S}$, and construct a Suffix Array $\mathcal{A}$ of $\mathcal{S}$. A suffix array [111] is a representation of a suffix tree [176] that can be constructed in linear time in $\|\mathcal{S}\|$ [82] and enables efficient computation of many substring queries; in particular, they allow us to identify duplicated training examples in linear time. Suffix arrays have the advantage over suffix trees in that they are 10–100× more memory efficient [111], requiring just 8 bytes per input token, though they are asymptotically less efficient for some query types. They have been used widely in NLP, such as for efficient TF-IDF computation [184] and document clustering [30].

The suffix array $\mathcal{A}$ for a sequence $\mathcal{S}$ is a lexicographically-ordered list of all suffixes contained in the sequence. Formally,

$$\mathcal{A}(\mathcal{S}) = \arg\text{sort all\_suffixes}(\mathcal{S})$$

For example, the suffixes of the sequence "banana" are ("banana", "anana", "nana" "ana", "na", "a") and so the suffix array is the sequence (6 4 2 1 5 3). In practice, we construct $\mathcal{S}$ from the BPE tokenization of the text (§4.4.7).

Substring matching

After constructing $\mathcal{A}$, it is straightforward to identify duplicated training examples. Suppose that the sequence s was repeated exactly twice in the training dataset $\mathcal{S}$ at positions i and j, that is, $\mathcal{S}_{i..i+|s|} = \mathcal{S}_{j..j+|s|}$. Then the indices i, j will occur adjacent to each other in the suffix array $\mathcal{A}$.

Finding all repeated sequences is thus a matter of linearly scanning the suffix array from beginning to end and looking for sequences $\mathcal{A}_i, \mathcal{A}_{i+1}$ that share a common prefix of at least some threshold length. Any satisfying sequences are recorded.

Setting a Threshold of Duplicates

One important question is how long a substring match must be before we ought to count it as a duplicate. In Figure **??**, we plot the frequency of substring matches within the four datasets we will consider. For each substring of length k, we compute the probability that there exists another sequence of length k identical to this one; formally:

$$m(k) = \Pr_{i \in [N]} \left[\exists j \neq i : \mathcal{S}_{i..i+k} = \mathcal{S}_{j..j+k} \right].$$

We choose 50 tokens as the threshold to be conservative: the "bend in the knee" occurs at 10 tokens, and manual inspection of length-25 matches found no false positives. We then doubled this value to have an exceptionally large margin for error.

Implementation

PARALLEL LINEAR TIME CONSTRUCTION. We build a parallelized linear time suffix array algorithm. As a building block, we make black-box use of the SA-IS algorithm for constructing a suffix array in linear time Nong et al. [124] and Ko and Aluru [85]. Unfortunately, this algorithm is not easily parallelized directly, so we introduce a simple divide and conquer approach to parallelizing the array construction.

We build our implementation in Rust and extend an existing suffix array library[15] with three modification. The first two are straightforward implementation differences: we modify the code to allow datasets larger than 4GB, and we remove the requirement that strings parse as valid UTF-8 sequences in favor of raw byte sequences. Our third change is more significant: we re-implement the algorithm so that we can stream the suffix array itself off disk.

PARALLEL PARTIAL SUFFIX ARRAY CONSTRUCTION. Our divide and conquer suffix array construction algorithm starts by partitioning the dataset into K different "splits" with SA-IS run over independently on each split in parallel. This algorithm still requires $O(N)$ work but runs in $O(N/K)$ wall-clock time. This gives us N separate suffix arrays $\mathcal{A}^i$.

Given two suffix arrays A_1 and A_2 for two sequences S_1 and S_2 it's not completely trivial to construct a single suffix array A for $S = S_1 \mid\mid S_2$ because of the boundary conditions. Instead, we don't build the data $S = S_1 \mid\mid S_2$ but rather let $S_1' = S_1 \mid\mid S_2[uptoK]$ for some K greater than the longest substring match. Then we build the arrays on S_1' and S_2. To merge the arrays together we can remove the items from the first array after index $|S_1|$ and merge-sort insert them into the second.

PARALLEL MERGE OF PARTIAL SUFFIX ARRAYS. We now merge these separate arrays together into a single suffix array $\mathcal{A}$, Consider the simpler case of two partial suffix arrays B and C that we would like to merge together. We can achieve this by letting $i = 0$ index B and $j = 0$ index C. Each iteration of the

---
15 https://github.com/BurntSushi/suffix

algorithm then pushes B_i into $\mathcal{A}$ if $S_{B_{i..}} < S_{C_i}$ and C_i otherwise, repeating until $i = |B| - 1$ and $j = |C| - 1$. To generalize to K splits, we need only replace the single comparison above with a min-heap requiring $O(\log K) \ll 10$ work on each iteration.

Observe that in the general case this algorithm is $O(Nm\log(K))$ where N is the length of the dataset, m is the average length of a prefix match, and K is the number of splits. It is therefore incorrect to call this algorithm linear time in the general case, for ours it is. Because the length of the longest match is bounded above by the length of the longest sequence, as long as the size of the dataset is independent of the length of the longest sequence in the dataset, this algorithm remains efficient.

Again, we can parallelize this operation among L simultaneous jobs (in practice we set $K = L$ as the number of threads on our machine). In the $K = 2$ case, job l processes $i \in [jN/L, (j+1)N/L]$, choosing the bounds of j by binary searching into C so that $S_{B_i} < S_{C_j} < S_{B_{j+1}}$. The case where $K > 2$ is identical except that we repeat this over all K partial suffix arrays.

Computational Analysis

We run our algorithm on a single VM on the cloud with 96 cores and 768GB of memory. Our algorithm is efficient, for example processing the Wiki-40B training set (3 million examples containing 4GB of text) in 2.3 minutes wall-clock time (2.1 CPU-hours of work). The 350GB C4 dataset takes under 12 hours (wall-clock) to build a suffix array; although we are still memory constrained and so this corresponds to $\sim$ 1000 CPU-hours. Once the suffix array has been constructed, it takes under an hour to deduplicate the C4 dataset.

Note that this algorithm still requires that the dataset itself fits in memory (so that we can efficiently index in arbitrary positions), but we do not need to fit the entire suffix array into memory. This is fortunate since our suffix array requires an 8× space overhead. For example, the suffix array for the 350GB C4 is 1.5TB.

Compared to the cost of training a language model on this dataset, the additional work required to deduplicate the training dataset is negligible.

Table 4.2: Qualitative examples of near-duplicates identified by NEARDUP from each dataset. The similarlity between documents is highlighted. Note the small interspersed differences that make exact duplicate matching less effective. Examples ending with "[...]" have been truncated for brevity. More data available in Appendix.

Dataset	Example	Near-Duplicate Example
Wiki-40B	\n_START_ARTICLE_\nHum Award for Most Impactful Character \n_START_SECTION_\nWinners and nominees\n_START_PARAGRAPH_\nIn the list below, winners are listed first in the colored row, followed by the other nominees. [...]	\n_START_ARTICLE_\nHum Award for Best Actor in a Negative Role \n_START_SECTION_\nWinners and nominees\n_START_PARAGRAPH_\nIn the list below, winners are listed first in the colored row, followed by the other nominees. [...]
LM1B	I left for California in 1979 and tracked Cleveland 's changes on trips back to visit my sisters .	I left for California in 1979 , and tracked Cleveland 's changes on trips back to visit my sisters .
C4	Affordable and convenient holiday flights take off from your departure country, "Canada". From May 2019 to October 2019, Condor flights to your dream destination will be roughly 6 a week! Book your Halifax (YHZ) - Basel (BSL) flight now, and look forward to your "Switzerland" destination!	Affordable and convenient holiday flights take off from your departure country, "USA". From April 2019 to October 2019, Condor flights to your dream destination will be roughly 7 a week! Book your Maui Kahului (OGG) - Dubrovnik (DBV) flight now, and look forward to your "Croatia" destination!

4.4.5 Method for Approximate Matching with MinHash

Overview

We also perform *approximate* deduplication based on matching entire examples. This method, which we call NEARDUP, is a good complement to the *exact* substring matching, especially for web crawl text, as it handles the very common case of documents being identical except for interspersed templated fields (such as the last row of Table 4.2).

MinHash [18] is an approximate matching algorithm widely used in large-scale deduplication tasks [168, 54, 64], including to deduplicate the training set for a large Chinese-language LM [188]. Given two documents x_i and x_j, the main idea is to represent each document by its respective set of n-grams d_i and d_j. We can then use hash functions to approximate the *Jaccard Index* [75]:

$$\text{Jaccard}(d_i, d_j) = {}^{|d_i \cap d_j|}/_{|d_i \cup d_j|} \tag{4.1}$$

If the Jaccard Index between d_i and d_j is sufficiently high, it is likely that documents are approximate matches of each other. To efficiently approximate the Jaccard index, MinHash constructs document signatures by sorting each of the n-grams via a hash function, and then keeping only the k smallest hashed n-grams. There are multiple ways to construct estimators of the Jaccard index from these kinds of signatures [36].

In our implementation, we use 5-grams and a signature of size 9,000. The probability that two documents are considered a potential match is

$$\Pr(d_i, d_j \mid \text{Jaccard}(d_i, d_j) = s_{i,j}) = 1 - (1 - s_{i,j}^b)^r \tag{4.2}$$

where $b = 20$ and $r = 450$ are user-settable parameters to control the strength of the filter.

For each pair of documents identified as a potential match, more computationally expensive similarity metrics can be employed as a subsequent filtering step. In particular, we identify two documents as duplicates if they are matched by the MinHash algorithm and their *edit similarity* is greater than 0.8. The edit similarity between token sequences x_i and x_j is defined as:

$$\text{EditSim}(x_i, x_j) = 1 - \frac{\text{EditDistance}(x_i, x_j)}{\max(|x_i|, |x_j|)} \tag{4.3}$$

To build clusters of similar documents, we construct a graph that has an edge between two documents if they are considered a match. Then, we use the method introduced in Łącki et al. [198] to identify connected components.

Implementation Details

For our MinHash based deduplication method, documents are first space tokenized, then each consecutive 5-gram is hashed using tabulation hashing. The set of these hashes is the signature for the document. For each element in a document's signature, the element is hashed using k other hash functions. The minimum hashed element for each of the k hash functions is stored. These minimum hashes are then partitioned into r buckets, with b hashes per bucket. These b hashes are augmented into a single value, then if two documents have the same value in at least one bucket, they'll be marked as a potential match. The probability that two documents are considered a potential match is equal to

$$\Pr(d_i, d_j \,|\, \text{Jaccard}(d_i, d_j) = s_{i,j}) = 1 - (1 - s_{i,j}^b)^r \tag{4.4}$$

where $s_{i,j}$ is the Jaccard index between the two documents. For document pairs that were identified as potential matches, we computed their actual Jaccard index, and if that was above 0.8, we computed their edit similarity. Document pairs with edit similarity higher than 0.8 were identified as duplicates. After some experimentation, we chose to use $b = 20$, and $r = 450$, so $k = 9,000$, so as to make sure a collision at the desired Jaccard index threshold of 0.8 had a high probability of occurring

We also tested an alternative configuration—filtering to document pairs with Jaccard index of at least 0.9 and edit similarity of at least 0.9. In this case, we used $b = 20$, $r = 40$, and $k = 800$. Figure **??** shows the histogram of Jaccard similarities and edit similarities for all document pairs which collided in min-hash space, for our chosen configuration (blue) and for the alternative configuration (orange). This allows us verify if the threshold chosen has few comparisons around the chosen threshold, then we've likely captured the majority of actual near duplicates above that threshold. To verify that yourself, look at the left hand tails of the distributions. Since both 0.8 and 0.9 begin to vanish at the same point (in spite of the fact that the two thresholds are optimized for accuracy around different thresholds), we feel comfortable saying that we're capturing the majority of actual near duplicates.

Computational Analysis

Let N be the number of documents and T be the maximal number of tokens in a document. Edit similarity has a worst case complexity of T^2, so the worst case complexity is

$$O(N + bk^2T^2N) = O(N) \tag{4.5}$$

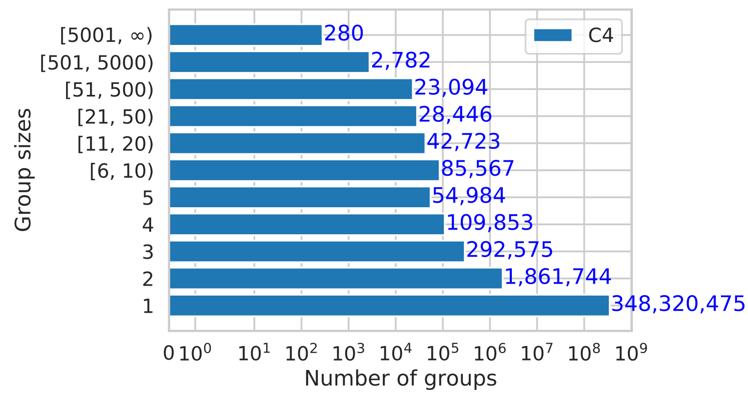


Figure 4.5: The distribution of near-duplicate cluster sizes from running NEARDUP on C4.

since b, k, and T are all $\ll N$. The left term is the complexity of grouping by the signatures, and the right represents the pathological worst case of all documents falling into the same B buckets.

The highly distributed NEARDUP implementation we employed is one used for large-scale production tasks at Google. On the English C4 dataset, the algorithm consumed approximately 41.5 kWh of energy. Note that our choices of k and b were designed to produce very high recall, and with different parameters, the algorithm could be made much more energy efficient while producing similar results.

4.4.6 Results

We deduplicate each of the four datasets with both of our two techniques. When text was duplicated across multiple data splits, we prioritized keeping a copy in the test or validation set and removing it from the train set.

Amount of Text Removed

With NEARDUP, we found that the web-scrape datasets contain between 3.04% (on C4) to 13.63% (on RealNews) near duplicates (Table 4.4). Near-duplicate text is much less common in Wiki-40B, forming only

Table 4.3: The fraction of tokens (note Table 4.4 reports the fraction of *examples*) identified by EXACTSUBSTR as part of an exact duplicate 50-token substring.

	% train examples with		% valid with
	dup in train	dup in valid	dup in train
C4	3.04%	1.59%	4.60%
RealNews	13.63%	1.25%	14.35%
LM1B	4.86%	0.07%	4.92%
Wiki40B	0.39%	0.26%	0.72%

Table 4.4: The fraction of examples identified by NEARDUP as near-duplicates.

	% train tokens with		% valid with
	dup in train	dup in valid	dup in train
C4	7.18%	0.75 %	1.38 %
RealNews	19.4 %	2.61 %	3.37 %
LM1B	0.76%	0.016%	0.019%
Wiki40B	2.76%	0.52 %	0.67 %

0.39% of the train set.[16] In C4, the majority (1.8M) of near-duplicate clusters consisted of just a single pair of examples that matched against each other, but there were 280 clusters with over 5,000 examples in them (Figure 4.5), including one cluster of size 250,933.

On average with EXACTSUBSTR, we remove more total content than with NEARDUP (despite EXACTSUBSTR not removing any examples outright)—for example removing 7.18% of the tokens in C4. The exception is LM1B, where EXACTSUBSTR removes 8× less data than NEARDUP. On investigation, we find this is due to the fact that LM1B documents are significantly shorter: 90% of all documents are under 50 tokens, and so are not even candidates for potential matches even if the entire sequence matched verbatim. We find that both NEARDUP and EXACTSUBSTR remove similar content—77% of the training examples that NEARDUP removes from C4 have at least one verbatim length-50 match found by EXACTSUBSTR.

16 Most duplicates we saw were automatically generated pages, such as the outcomes of sports games. This shows the strength of manual curation for creating high-quality datasets.

Table 4.5: On the left, we show the URLs that had the greatest proportion of examples marked as near-duplicates by NEARDUP(filtered to URLs which occurred at least 10 times). On the right, we show the 20 most frequent URLs in C4 for which all examples were marked as near-duplicates by NEARDUP.

RealNews Url	# Total	Frac Dups	C4 Url	# Total	Frac Dups
medicalnewstoday.com.	12	1.00	hairtechkearney.com	4883	1
dodbuzz.com	301	0.99	keywordsking.com	1786	1
undertheradar.military.com	187	0.97	sydneysitalianfruitshops.online	1178	1
q.usatoday.com	33	0.94	moewiki.usamimi.info	1001	1
ad-test.thirdage.com	354	0.94	swarovskijewelryoutlet.org	984	1
amp.nymag.com	15	0.93	forzadurto.org	980	1
citizenwire.com	1022	0.93	producerati.com	971	1
paycheck-chronicles.military.com	363	0.92	sourceryforge.org	908	1
product-reviews.net	73403	0.92	heavenz-kitchen.com	876	1
kitup.military.com	196	0.92	little-eclipse.com	822	1
gcaptain.com	33903	0.92	walops.com	819	1
dev.screenrant.com	70	0.91	16thstlaunderland.com	713	1
live.swissinfo.ch	66	0.91	theroyalstarinfo.com	696	1
news.theepochtimes.com	82	0.87	code4kt.com	684	1
opinion.toledoblade.com	986	0.87	nflfalconsjerseys.us	682	1
cdn.moneytalksnews.com	121	0.86	quiltingbeeshop.com	676	1
amp.fox23.com	14	0.86	ulifeinsurancemiami.com	675	1
sales.rollingstone.com	20	0.85	wowkeyword.com	673	1
ftp.screenrant.com	20	0.85	taspetro.com	671	1

4.4.7 Properties of Duplicated Text

While the authors of both RealNews and C4 explicitly attempted deduplication during dataset construction, the methods were insufficient to capture the more subtle types of duplicate text commonly found on the internet. In C4 and Wiki-40B, we qualitatively observe that much of the text identified as near-duplicated is computer-generated. The text is identical except for the names of places, businesses, products, dates, and so on. Because these examples frequently differ by just a few words at a time, deduplication strategies relying on exact string matching would fail to identify a match. Example duplicate pairs from each dataset can be found in Table 4.2. Table 4.5 shows the URLs had the largest proportion of examples identified by NEARDUP as near-duplicates. For C4, these tend to be websites that sell many similar products and thus have a large amount of templated text. For RealNews, content aggregators seem especially common.

For RealNews and LM1B, derived from news sites, we observe that many near-duplicates occur because the same news article appears on multiple news sites with slightly different formatting. For example, in LM1B, there is one example that starts "*MINEOLA , N.Y. - New York officials say* [...]" and another that starts "*(AP) - New York officials say* [...]". The two examples are otherwise identical.

Train / Test Set Leakage

Both deduplication methods identify overlap between the train set and the validation set (Table 4.4). For example, 4.6% of the C4 validation set and 14.4% of the RealNews validation set examples had an approximate duplicate in their respective training sets. Such duplication is problematic since it could cause evaluation metrics to be unfairly inflated for models that are better at memorizing their train sets. We evaluate the effect of this leakage on publicly released models in Section 4.4.7.

Impact on Trained Models

. We trained 1.5B parameter "XL", decoder-only, Transformer-based language models similar to GPT-2, on C4-ORIGINAL, C4-NEARDUP, and C4-EXACTSUBSTR, respectively. We use the T5 codebase and model architecture from Raffel et al. [132], and each model was trained for about two epochs on its respective dataset. To better understand the amount of variance in the perplexities of trained models, we also trained three different random seeds of the 110M parameter “base" model for each of the above three datasets—for a total of nine base-sized models.

For all experiments, we used a Byte Pair Encoding (BPE) vocabulary trained on C4-NEARDUP with a budget of 50K tokens, which resulted in a vocabulary the same size as GPT-2’s. We trained with a maximum sequence length of 512 tokens (for longer documents, we randomly extracted subsequences of this length.) Each model was trained for about two epochs. Since both C4-ORIGINAL and C4-EXACTSUBSTR contain approximately 365M examples, we performed 152K steps with a batch size of 4800 (or approximately 2 epochs). C4-NEARDUP contains approximately 350M examples, we performed 146K steps (or approximately 2 epochs). On a 128-core TPU v3 pod slice, XL models trained on C4-ORIGINAL and C4-EXACTSUBSTR took approximately 131 hours (5.5 days) to train, while the XL model trained on C4-NEARDUP took approximately 126 hours to train. Like T5, models were trained with the Adafactor optimizer [147]. A constant learning rate of 0.01 was used for the base models and 0.001 for the XL models.

The 1.5B parameter XL models had 24 layers, each with 32 attention heads. The model embedding size was 2,048, the feed forward layers had a hidden size of 5,120, and the key/value dimension size for the attention heads 64. The 110M parameter base models had 12 layers, each with 12 attention heads. The model embedding size was 768, the feed forward layers had a hidden size of 2,048, and the key/value dimension size for the attention heads 64.

MODEL PERPLEXITY We computed the perplexity of our trained models on the validation sets of LM1B and Wiki-40B, and on subsets of the C4 validation set (Figure 4.6). For the base size, we observe that all models have similar perplexity on the original C4 validation set and on validation set examples that were identified as unique (no near-duplicate in either train or validation). However, both models trained on deduplicated data have significantly higher perplexity on validation set examples that have duplicates in the training set than the model trained on the original C4. EXACTSUBSTR-deduplicated results in higher perplexity than

Table 4.6: When generating 100k sequences with no prompting, over 1% of the tokens emitted from a model trained on the original dataset are part of a 50-token long sequence copied directly from the training dataset. This drops to 0.1% for the deduplicated datasets.

Model	1 Epoch	2 Epochs
XL-ORIGINAL	1.926%	1.571%
XL-NEARDUP	0.189%	0.264%
XL-EXACTSUBSTR	0.138%	0.168%

NEARDUP-deduplicated. These trends holds true for the XL sized model as well. While this may suggest EXACTSUBSTR duplication results in models least overfit on the train set, note that both of these techniques have used separate duplicate thresholds and a different choice of thresholds could change the results.

When evaluating on the validation sets of LM1B and Wiki-40B, we found that models trained on NEARDUP-deduplicated C4 consistently achieved lowest perplexity. EXACTSUBSTR deduplication decreases perplexity of the XL model by almost 3 points perplexity on Wiki-40B which is much larger than the variation of about 1 point perplexity we observed in the base models. This is despite seeing fewer tokens of training data overall.

Lastly, we note all our XL models achieved <35 perplexity on LM1B, which is less than the 42.16 perplexity reported for the 1.5B GPT-2 using a vocabulary the same size as ours.

Impact on Generated Text

Data duplication has the effect of biasing the trained LM towards particular types of examples. This can contribute to a lower diversity of generations, and increased likelihood that the generated content is copied from the training data [27]. For our generation experiments, we use top-k random sampling with $k = 50$ and experiment with prompted and unprompted generation.

We first evaluate memorization tendencies in the case where the model is asked to generate text without any prompt sequence. We generate 100,000 samples, each up to 512 tokens in length. For each generated token, we say the token is memorized if it is part of a 50-token substring that is exactly contained in the training data. On XL-ORIGINAL, over 1% of the generated tokens belong to memorized sub-sequences (see Table 4.6). This is $\sim 10\times$ more memorization than XL-ExactSubstr or XL-NearDup. Some example subsequences that were copied verbatim from the train set can be found in Table 4.7.

In most real use cases, language model generation is controlled by providing a prompt for the model to continue. We experiment with four possible prompt sources: training examples identified by ExactSubstr as having near-duplicates in the train set (train dup), training examples identified as unique (train unique), validation set examples with a near-duplicate in the train set (valid in train), and validation set examples identified as unique across all splits (valid unique). We select the first 32 tokens of each example as the prompt, which means we can evaluate the fraction of generations which are near-duplicates with the ground-truth continuation for the prompt. Figure 4.8 shows the proportion of generations which meet this requirement, while Figure 4.7 shows the distribution in edit similarities between the generations and ground-truth continuations. When the prompt comes from duplicate examples in the train set, XL-Original reproduces the groundtruth continuation over 40% of the time. XL-ExactSubstr and XL-NearDup still copy the groundtruth more often when the prompt comes from a duplicate example than when the prompt comes from a unique example, suggesting that more stringent deduplication may be necessary to remove memorization tendencies entirely.

Impact on Existing Models

Train-test leakage does not just impact models trained on C4. Table 4.9 shows that the presence of near-duplicates of the evaluation set in the train set has a significant impact on model perplexity for two standard models: Transformer-XL [39], which was trained on LM1B, and GROVER [187], which was trained on RealNews. For Transformer XL, the perplexity halves on examples identified as near-duplicates. For GROVER, the difference, though not quite as stark, is present in both model sizes considered.

Existing models also suffer from the problem of generating text from their train sets. We find that 1.38% of the tokens in the official release of 25k GROVER-Mega outputs are part of verbatim matches in RealNews of at least length 50. Likewise, more than 5% of the tokens in ~200k sequences outputted by GPT-Neo 1.3B [15] are part of a 50 token matches of its training data, the Pile [56].

4.4.8 Discussion

The focus of this paper is on the datasets used to train language models. While recent work focused on documenting the potential harms that could arise from problematic datasets [14, 58], less work has been done to quantitatively analyze properties of real language modelling datasets, like Dodge et al. [43] has done for C4. Our paper provides analysis on one particular axis, that of data duplication.

Our experiments measured what could be quantified: the amount of duplicate content in common datasets, the effect of deduplication on trained model perplexity, and the reduction of memorized content in trained models through deduplication. We do not focus on the nature of the data being removed by deduplication or memorized by LMs.

Privacy is an important subject for future work, as memorized training data has significant privacy consequences. By this, we mean the standard privacy definition that a model should not reveal anything particular to the specific dataset it was trained on, as opposed to another training dataset from a similar distribution [150].[17] Training on standard datasets that have not yet been deduplicated results in models that are particularly sensitive to examples that happened to be repeated multiple times, and this has negative privacy implications. For instance, it could violate a person's expectations of privacy if their publicly available personal data appeared in a different, surprising context. Downstream applications of LMs, such as the game AI Dungeon[18], should also not output memorized content like adverts for real products.

We stress that in our experiments, we do not distinguish between undesired memorized text (such as phone numbers), innocuous memorized text (common phrases), and text we may want to be memorized (such as a quote by a public figure), and instead treat all instances of the LM generating text that closely matches the training set as problematic. While we qualitatively observed that much of the identified memorized content was relatively innocuous, a more systematic study of the risks associated with the detected memorization was beyond the scope of this work.

---

17 Another interpretation of privacy focuses on the sensitivity of the data involved, when a model is trained on and able to reproduce personal identifiers or other forms of "private data." Our definition is more expansive.

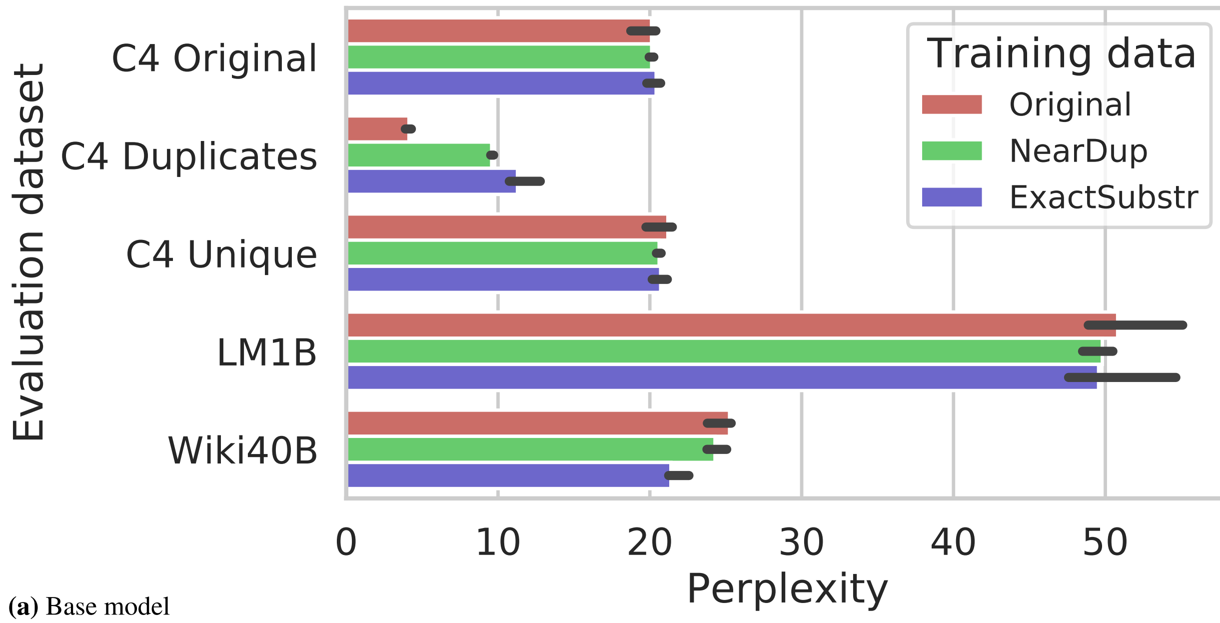
18 https://play.aidungeon.io/

We also do not investigate the negative consequences of deduplication. Some language tasks explicitly require memorization, like document retrieval or closed-book question answering. Also, text that gives attribution is often duplicated across documents, so removing duplicate substrings could correspond to removing *just* the attribution, which could result in models that learn the content without its attached attribution. Deduplication is also not sufficient to remove privacy-sensitive data like bank passwords and medical records which should never be used in training data.

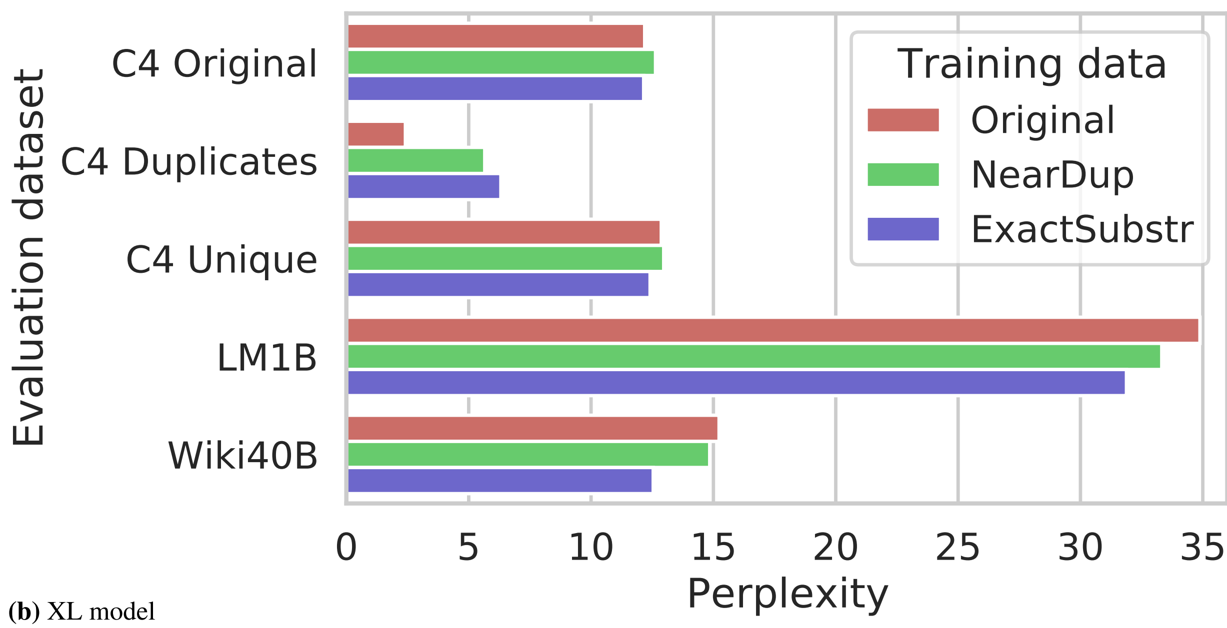
Ultimately, whether memorization is a desired property of a language model, or else risky and unwanted, depends on the nature of the text that has been memorized and on the downstream applications of the trained model. However, since the trend has been towards creating datasets and models that are application-agnostic, we encourage researchers to think carefully about the limitations of the data collected and the how the model's intended usage constrains what should be part of the training set. Developing techniques to memorize or forget specific sequences depending on the end application is a promising research direction.

We encourage future language model research to perform dataset deduplication, either by training on the deduplicated datasets we release, using the deduplication tools we release, or following our approach to deduplicate datasets with new tools.

The exact technique used to perform deduplication is less important than performing stringent deduplication in the first place. On the whole, deduplication does not harm, and sometimes improves, model perplexity, despite the fact that the deduplicated datasets are smaller and faster to train on. It is especially important that there are no duplicates between the training and testing sets, because overlap here explicitly encourages selecting models that memorize the training data. Lastly, deduplication helps to reduce some of the privacy concerns around LMs memorizing their training data.



(a) Base model



(b) XL model

Figure 4.6: Impact of deduplicating the training set on validation perplexity. In **(a)**, we plot the results from T5 base (110M parameters) across three training runs with different random initializations. The black bar represent the lowest perplexity to the highest perplexity, and the colored bar the median perplexity. In **(b)**, we plot the results from T5 XL (1.5B parameters). For C4, we evaluate on *C4 Original*, the original validation set; *C4 Unique*, a subset of the validation set identified by NearDup as having zero matches across C4; and *C4 Duplicates*, a subset of the validation set identified by NearDup as having a match in the C4 train set.

Text	Freq in C4
HD wallpaper. This wallpaper was upload at April 19, 2019 upload by admin in.You can download it in your computer by clicking resolution image in Download by size:. Don't forget to rate and comment if you interest with this wallpaper.	40,340
to the address posted below. Include our failure information form,a packing slip with your Company name, contact person, and Email address or phone number. Upon receipt of your repair, we\'ll inspect it and then contact you with a quote or evaluation notice. Normal turn aro und for repair is 5 to 7 business days, with "Rush Repair" available.	5,900
is a great place to begin your search. Whether you are a first-time home buyer or you are already familiar with the home buying process, you can be assured that you have the best tools and the perfect agent available to help with your	5,358
pics at these awesome group starting P letter. Desktop wallpapers were first introduced way back in the 1980s and have gained immense popularity since then. It is possible to come across more than 80 million sites on the web offering some sort of wallpaper.	848
flowers will let them know you're thinking of them and wishing them well. Cheerful yellow flowers bring their own sunshine and will get right to work on lifting spirits, and a colorful vase will bring loads of smiles to friends and visitors! Get Well flower arrangements from	479
our premier 24 hour emergency* plumbing and heating solutions. We realise that when your heating fails or pipes and drains leak it can cause havoc with your routine and even cause damage to your property. When a plumbing problem occurs that requires an immediate response we provide qualified local plumbers throughout	56
is to remove all images that violate copyrights. Please contact us to request that images be removed or to assign proper credit. The images displayed on this site may be used for Free or educational purposes only. If you would like to use any of the images displayed on this site for any other purpose, please obtain permission from the owner. www.	48
list of fishing locations, providing interactive maps that show each location's GPS coordinates, nearby facilities (like restaurants, gas stations, marinas and fishing shops), their current and forecasted weather and, if available, their water conditions.\nFind any of the 8	5
. Dyer, Ph.D., is an internationally renowned author and speaker in the field of self-development. He's the author of 30 books, has created many audio programs and videos, and has appeared on thousands of television and radio shows.	5

Table 4.7: A selection of substrings identified by ExactSubstr as being in C4 multiple times. The number of times this exact substring occurs in C4 is also given.

Table 4.8: A selection of substrings generated by XL-ORIGINAL with no prompting (and top-k with k=50) that were identified by EXACTSUBSTR as being in C4 multiple times. The number of times each substring was found in C4 is given. We observe that most memorized generations tend to be from advertisements.

Generated Text	Freq in C4
, you'll need to be knowledgeable to make the very best decisions. We will make sure you know what can be expected. We take the surprises from the picture by giving accurate and thorough information. You can start by talking about your task with our client service staff when you dial 888-353-1299. We'll address all of your questions and arrange the initial meeting. We work closely with you through the whole project, and our team can show up promptly and prepared.	5,497
then Waterside Lodge are well equipped for the task. Our fully equipped family sized lodges offer a comfortable luxurious stay for a fantastic price, giving you beautiful views of the lakes and the surrounding countryside. Offering luxurious self-catering holidays in our fully featured Scandinavian holiday lodges. Perfectly located to explore the beaches, coastline. All of our lodges are sized for 6 people and are furnished to the highest standards to ensure you have a stay like no other. At Waterside Lodge the stay itself is only half of the package, Waterside lodge is situated closely to the Heritage Coast which makes our lodges the perfect stay for anyone wanting to get away and have a relaxing countryside break from the city. Whilst you stay with us be sure to take advantage of all the activities Waterside Lodge has to offer. Such as the use of our on-site fishing lakes for the keen fisherman, free internet access, outside relaxation areas, comfortable lounges and much more.	571
you are only looking to find rent to own homes in your city or are open to exploring all kinds of rent to own home listings, our database does it all. One of the best aspects of iRentToOwn.com is that, besides options to rent to buy a house, it has numerous other categories of home sale options. These include bank foreclosure homes, pre-foreclosure homes, short sales, HUD/government foreclosures, auction homes and owner-financing/FSBO (For Sale By Owner) homes. With help from the convenient search features offered by our site, shoppers are able to find their ideal lease to own home, real estate company, and more in South	51
, IL employs journeyman as licensed to work by themselves, without direct supervision, installing wiring, outlets and fixtures. Our journeyman also does service work, troubleshooting when a breaker fails or a light stops working. Our journeyman does not offer permits that must be issued by our master. Our journeyman follows our master's plans and directions. Our journeyman's responsibilities will vary based on the work that needs to be done. Our journeymen are skilled with residential, commercial and industrial installations and repairs.ust work from six years as an apprentice, under direct supervision of our master, and pass a journeyman test. This person also must have some classroom education on the National Electrical Code and fundamental electricity in a technical school a program affiliated with the National Joint Apprenticeship Training Council. Journeyman training combines hands-on work with education on basic electricity.	6
combustion process of a petrol engine is never perfect. Dangerous gases, such as nitrogen oxide, carbon monoxide and hydrocarbons will arise and it is the job of the catalytic converter to reduce these to safer emissions. These cat converters can fail by becoming clogged, or if the engine has bad exhaust valves or the plugs fail, causing unburned fuel to overheat the converter. Mettam's Mufflers can resolve these issues with your Karr	5
,ANDREW Find the ancestral town: Many a researcher is stuck behind records that say, BIRTH-PLACE: IRELAND without saying where in Ireland, or whatever other country. Remember that your immigrant ancestor's siblings probably were born in the same ancestral town, so check all o f their records, too. Around 1900, the Roman Catholic churches reported marriages to the churches where the persons were baptised, and before the wedding, they would require a baptismal certificate from that church, without marriage notations, to make sure that the persons were no t already married, ordained, or whatever, and were free to marry. Do check the Catholic records especially for ex loco and the home town. If your ancestor's sister had a daughter who generated a marriage or death record saying, MOTHER'S BIRTHPLACE: and the exact town, then y	2

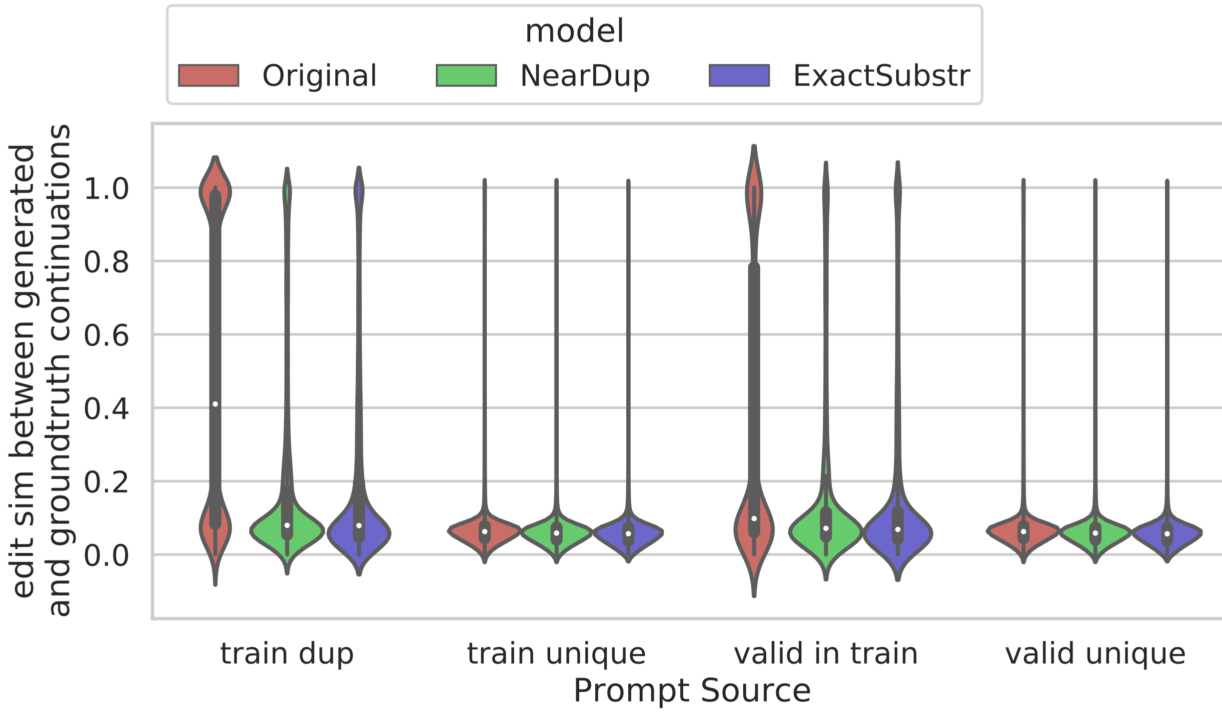


Figure 4.7: Memorized continuations distribution

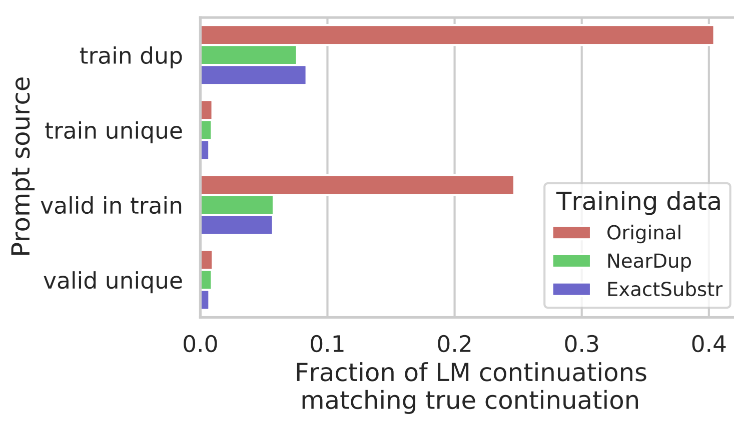


Figure 4.8: The proportion of generations which have edit similarity above 0.8 with the groundtruth continuation when using the LM to generate continuations for 32-token prompts identified by NEARDUP as either duplicated or unique.

Table 4.9: For each model, the perplexity of the official validation set (*Orig*), valid set examples which were identified by NEARDUP as matches of train set examples (*Dups*), and valid set examples identified by NEARDUP as unique (*Unique*). Due to the size of the RealNews validation set, we evaluated on only the first 25k examples meeting each condition.

Model	Dataset	Orig	Dups	Unique
Transformer-XL	LM1B	21.77	10.11	23.58
GROVER-Base	RealNews	15.44	13.77	15.73
GROVER-XL	RealNews	9.15	7.68	9.45

5 ENABLING APPLICATIONS IN CREATIVE WRITING

One application where NLG has considerable potential is in the development of tools for creative writing. AI-assisted creative writing is an attractive testbed NLG systems because ideation tools are already part of writers' arsenal, and mistakes like hallucinating false facts are less problematic in fiction than in domains like automatic news summarization, where faithfulness to the real world is crucial. In addition, writers have been grappling with the concept of sentient, human-like machines for at least as long as computer scientists have.

In this chapter, I describe work I have done toward bridging the gap between what most language models do by default (predict a continuation for a prompt) and the operations writers actually would want. First, I will describe efforts to capture longer-term coherence by building a language model that operates over sentences rather than other sub-words. Second, I will show how existing neural networks can be modified to support fill-in-the-blank style tasks in addition to the more common paradigm of continuation. Filling in the blank is a common control that is requested by writers. Third, I will present a recipe for performing sentence style transfer into arbitrary styles–such as rewriting text to be more Shakespearean, metaphorical, or meladramatic–without any exemplars of the task or task-specific model training.

To test out how these and other NLG-based tools can be used in practice, we built Wordcraft, a word processor augmented with a variety of "smart" writing controls and suggestion tools. I will end by describing the features of Wordcraft and the results of a user study conducted with the tool.

5.1 MOTIVATION

Artificial intelligence systems which can write stories have been a goal of researchers since the early days of computing. However, much of the early work in this area was focused on building systems which could produce an entire story from scratch given an initial set of constraints.

Rather than focusing on developing NLG systems which produce entire stories, my research in this area focuses on enabling better human-AI collaboration.

5.2 TOWARDS SENTENCE-LEVEL LANGUAGE MODELS

5.3 MODELS FOR INFILLING TEXT

5.3.1 Motivation

Natural language generation systems are increasingly being incorporated into applications where a human writer and an AI jointly collaborate to construct text. Wordcraft, The AI-assited text processor I describe in Section **??** is one such application. Another is Storium, where players of a writing game, have the option to accept suggestions from a natural langueg generatin system [3]. There are also more practical domain such as email composition assistance and code synthesis [24, 179, 10]. Many of these applications are limited to generating text at the end of what has been written so far. This is because both historical language models (LMs) and state-of-the-art neural LMs are typically designed to produce text by repeatedly predicting the next word in a sequence given the previous words. However, there is a need for more powerful interactive tools which enable writers to solicit insertions at any chosen position within the existing text, a task variously referred to as fill in the blank (FITB), infilling, or the Cloze task [158]. For example, a creative writer might

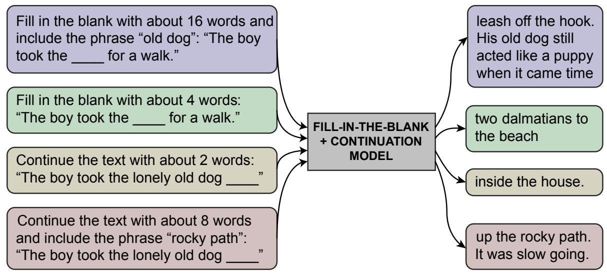


Figure 5.1: A single model that can handle a variety of related writing tasks is more efficient than separate models per task.

want a tool which can insert a description of a place or character, and a programmer might want a system that can fill in a method in the middle of their code.

Most prior work tackling FITB consider it a separate task from continuation, one to be specifically optimized for, for example training a custom model from scratch [73, 194, 117], finetuning a model trained originally for continuation [45], or using a combination of pre-trained models [71]. Having separate trained models for FITB and for continuation is inefficient for downstream applications where maintaining multiple neural networks can b e prohibitive.

Any model that can do FITB can be made to do continuation simply by placing the blank at the end of the input. Thus, in this section, we describe how models trained on FITB can be employed effectively for both infilling and continuation operations. We show how T5 [132], one of the most popular pre-trained models, can reasonably handle both tasks, as it was pre-trained with a FITB-like objective. Finetuning T5 further improves its ability and also allows for the incorporation of controllability of generation length and word choice.

5.3.2 Supporting FITB and Continuation

We define filling in the blank as the task of predicting text to replace a single missing span, usually demarcated with a special token, in an input text passage. (Some prior work considers inputs with multiple blanks, but inserting text at one position at a time better matches the kinds of edits humans do.) We define continuation in the traditional language modeling sense–predicting the next token in a sequence given only the previous

Table 5.1: Examples of the finetuning objectives. “8" is the approximate length in words of the target sequence. During finetuning, about 25% of training examples took each of these formats.

Example Type	Input	Target
C4FILLBLANK no goal	fill: I love avocados. I ate a sandwich covered in them. _8_ I talked to my doctor about it later. It turned out I was allergic to avocados.	After I ate it, my mouth was itchy and tingly.
C4FILLBLANK with goal	fill: I love avocados. I ate a sandwich covered in them. _8_ I talked to my doctor about it later. It turned out I was allergic to avocados. **G**oal: mouth was itchy	After I ate it, my mouth was itchy and tingly.
C4FILLBLANK no goal	fill: I love avocados. I ate a sandwich covered in them. After I ate it, my mouth was itchy and tingly. I talked to my doctor about it later. _8_	It turned out I was allergic to avocados.
C4FILLEND with goal	fill: I love avocados. I ate a sandwich covered in them. After I ate it, my mouth was itchy and tingly. I talked to my doctor about it later. _8_ **G**oal: allergic to	It turned out I was allergic to avocados.

tokens. Donahue et al. [45] discuss how language modeling is a special case of infilling, and they use this as justification to finetune a continuation-based language model to do infilling. However, we argue that if continuation is a subtask of infilling, it makes more sense to go in the opposite direction: prioritize a model which can do infilling and check that it achieves satisfactory performance at continuation.

T5 is a model pre-trained with a “span corruption” objective very similar to FITB; the model is asked to reconstruct the missing text after random sub-sequences of the input are replaced with special identifiers. Thus, a pre-trained T5 model can be used without any further training to do both continuation and infilling by appropriately choosing text to mask out. The encoder-decoder architecture of T5 is also more conducive to FITB than decoder-only architectures like GPT-2 [131] which are typically used for continuation-based language models. This is because the attention mechanism in encoder-decoder architectures allows the context on the left side of the blank to attend to the context on the right, while decoder-only architectures only support masked attention (each token can only attend to the positions to its left).

Even though T5’s pre-training objective was a form of FITB, finetuning is still advantageous. For one, our definition of FITB only includes a single masked out substring, not multiple, so finetuning improves alignment with the goal task. Finetuning also allows us to incorporate additional conditioning signals not supported by the pre-trained T5, such as being able to specify the desired length of the generated text or specify words that ought to be included in the blank, a task we refer to as “goal conditioning." Length control, which comes by default in a traditional language model by simply sampling more or fewer tokens, is particularly necessary for FITB, where the end of the generation must fit seamlessly with the text to its right.

The biggest language models available today were largely trained in the continuation rather than the FITB paradigm [21, 15]. Since our primary goal is to have a single model for both tasks, we also address the question: if a continuation-trained model is big enough, can it handle FITB without the need for finetuning? Few-shot learning with large language models, as popularized by Brown et al. [21], has had success on many tasks in NLP. We try out this approach for FITB by designing a few-shot prompt containing several demonstrations of the FITB task, formulated in a similar “infilling by language modelling" style as Donahue et al. [45].

5.3.3 Experimental Setup

For all primary experiments, we use the 800M parameter v1.1 ‘large’ model. We also show soome addditional results comparing against the 3B parameter ‘XL’ T5 model. To finetune T5 for FITB, we construct training examples from documents by first partitioning the document text into a left context, gap, and right context. The input sequence is then the left and right contexts concatenated with textual representations of the additional conditioning signals. The target sequence is the true text for the blank. This formulation easily supports continuation, as the blank can be deliberately placed at the end (i.e., providing no right context). Documents are drawn from C4, the same dataset T5 was pre-trained on. Documents are split into word sequences, and these are then randomly truncated to be between 256-512 words long. A substring of between 1 and 64 words is selected to be blanked out. For half of the training examples the blank is randomly selected, and for the other half it is always placed at the end. To support length conditioning, we follow Roberts and Raffel [139] and include a bucketed version of the target length as part of the blank. To support goal conditioning, for half the examples, a random substring of up to half the words of the target is appended to the end of the input. Examples are shown in Table 5.1.

Table 5.2: Perplexity of evaluation sets according to LLM when the blank has been filled with approaches involving no fine-tuning (top), finetuned approaches (middle), and the groundtruth (bottom). Lower values indicate that the text was considered more fluent by the LLM.

	C4FILL BLANK	RWPFILL MIDDLE	ROCFILL BLANK
Few-shot LLM	14.14	19.48	18.21
Pre-trained T5	10.38	14.08	22.62
Finetuned T5	10.33	14.08	20.47
Donahue et al. [45]	N/A	N/A	23.28
Groundtruth	9.41	12.99	16.90

We compare T5 against a state-of-the-art 137B parameter decoder-only language model (LLM) trained explicitly for continuation and used successfully for few-shot learning in other domains [10, 136]. This model is used (1) as a standard continuation model, prompting with only the left context of an example; and (2) in a few-shot learning paradigm.

We evaluate continuation and FITB on C4 as well as two story writing datasets, as creative writing assistant applications are one of the key areas we expect to benefit from multi-task models [35]. Reddit Writing Prompts (RWP) is a corpus of stories from the 'r/WritingPrompts' sub-Reddit [49], and we construct validation sets RWPFILLBLANK and RWPFILLEND using the same method described in the previous section. C4 and RWP validation sets are capped to 5,000 examples. ROC Stories (ROC) is a crowd-sourced dataset of five-sentence commonsense stories [118]. For ROC Stories, the 2018 validation set is used to construct ROCFILLMIDDLE, where the middle sentence of each story is blanked out, and ROCFILLEND, where the last sentence is blanked out. Unless otherwise noted, all evaluation is done without goal conditioning and uses random sampling with top-k=50 as the decoding strategy. Example generations for all evaluation sets can be found at https://bit.ly/2U0Ixxa.

Table 5.3: Perplexity of continuation-based evaluation sets when a continuation has been generated using approaches with no finetuning (top) and two settings of finetuning T5 (middle).

	C4FILL END	RWPFILL END	ROCFILL END
Pre-trained T5	10.09	13.51	21.71
T5 FILLBLANKCONT	10.04	13.74	19.60
T5 LM-ADAPTION	10.06	13.71	19.68
Groundtruth	9.41	12.99	16.90

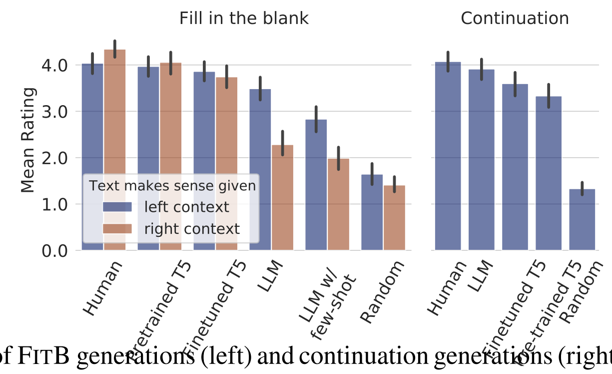


Figure 5.2: Human ratings of FITB generations (left) and continuation generations (right). Error bars are 95% confidence intervals.

5.3.4 Results

Difficulty in Developing a Few-Shot Prompt for the Infilling Task

Filling in a blank seems like a task that ought to be easy to accomplish with few-shot learning techniques. Training data for large language models often contains fill-in-the-blank style examples, as school lessons with cloze-style questions are relatively common on the internet. Furthermore, infilling ought to be an easier task tha continuation since there is more information available for the model to base its prediction on. However, after conducting a large-scale study of many possible few-shot prompts, we found that this technique fell short for the fill-in-the-blank task.

Choosing appropriate examples for a few-shot prompt is very challenging because task performance is often sensitive to minor changes in prompt design [193]. We experimented with prompts randomly selected from

Table 5.4: Accuracy of models finetuned on FILLBLANKCONT at correctly using provided length and goal conditioning signals.

Finetuned T5	Context	Length
C4FILLBLANK	0.860	0.877
RWPFILLBLANK	0.797	0.881
C4FILLEND	0.858	0.775
RWPFILLEND	0.791	0.746

the C4, Reddit Writing Prompts, and ROC Stories training sets, as well as prompts consisting of examples we hand-wrote with the goal of story-writing in mind. For each prompt source, we randomly generated five possible prompts, each with three examples, using the method descrribed in Section **??**. To simplify the task, we conditioned on desired length but did not include goal conditioning.

An example prompt is shown in Figure **??**. When choosing random few-shot prompts from the dataset train sets, in order to keep the few-shot prompt text within the 512-token context length limit of model we used for inference, we only considered examples that contained 100 or fewer tokens, so that the max length of the few-shot prompt was no more than 300 tokens. This left 212 tokens for the text of the actual example we were interested in performing the FITB task on. For each evaluation set, examples with inputs longer than 212 tokens were excluded from analysis. For our hand-written prompt, we wrote the 7 examples shown in Table **??**. We generated 5 possible prompts by randomly subsampling 3 examples out of these 7.

Table 5.5 shows the perplexity of the generations from each few-shot prompt. We note that even leaving room for 212 tokens worth of context text, some evaluation examples did not fit in the prompt length, and these examples were skipped when doing this analysis. Figure **??** shows a histogram of the fraction of validation set examples that remained for each few-shot prompt after the too-long examples were filtered out. Based on these results, we chose to include in human evaluation the best few-shot prompt from from ROCFILLMIDDLEand the best few-shot prompt from C4FILLBLANK. Figure 5.2 in the main paper shows the result from the C4FILLBLANKfew-shot prompt, whose outputs were rated slightly higher by human annotators.

Unfortunately, it is computationally impossible to conduct an exhaustive search of all possible example choices and prompt formats. While none of the prompts we tried led to strong performance at the fill-in-the-blank task, we cannot rule out the possibility there might exist a prompt we did not test for which this performance might have been significantly better. For example, we did not conduct formal experiments to systematically vary the prompt wording/formatting shown in Figure **??**. We can conclude that the process of finding an ideal prompt requires time-consuming trial-and-error and is quite difficult!

Table 5.5: Perplexity of evaluation sets when the blank has been filled in using LLM with few-shot prompting (top) and our best fine-tuned T5 model ((bottom). Among the few-shot results, the best method for each dataset is bolded, as well as methods within one standard error.

Few-shot source:	C4FILL BLANK	ROCFILL MIDDLE	RWPFILL BLANK	RWPFILL BLANK-Sent
C4FILLBLANK	15.67	19.72	**19.65**	**16.82**
ROCFILLMIDDLE	**14.14**	19.61	**19.48**	**16.36**
RWPFILLBLANK	24.39	20.29	32.33	28.13
RWPFILLBLANK-Sent	18.91	**18.21**	24.44	19.87
FS CUSTOM	17.98	19.80	21.72	18.38
Finetuned T5 XL	9.99	19.00	13.64	10.03
Finetuned T5 Large	10.33	20.47	14.08	10.37

Automatic Evaluation

We measure the fluency of proposed generations by evaluating the perplexity of each dataset's examples when the predicted text is placed in the blank [45]. We use the LLM to measure perplexity[19]. The results are shown in Table 5.2. We see that the LLM struggles to generate fluent infills, even when used in a few-shot setting. The only exception to this is ROC Stories, a dataset with fairly simplistic, predictable language. Finetuning T5 does not result in significantly improved fluency over the pre-trained model except on ROC Stories. Lastly, for ROC Stories, we compare against Donahue et al. [45]'s finetuned GPT-2 small, which yielded less fluent

19 Note, since this is the same model being used for generation for our continuation baseline, this metric may be biased.

predictions. Table 5.3 shows a similar analysis on our continuation-style datasets. Both T5-based models achieve roughly the same fluency.

Human Evaluation

Human evaluation was conducted on 70 examples, 35 from RwpFillBlank and 35 from RwpFillEnd, with examples about evenly distributed across length buckets. For RwpFillBlank evaluation tasks, the rater was presented an input context and several possible sequences that could go in the blank. They were asked to rate each sequence first, on how well it fit the text before it, and second, on how well it fit with the text following it, according to a 5-point slider For RwpFillBlank, the task was almost the same, except that the rater was presented only a left context and asked to rate how well it continued the prompt. A screenshot of the Human Intelligence Task (HIT) used for annotations is shown in Figure 5.3. Workers were paid originally paid $1.85 per HIT, but since the average HIT duration ended up being 15 minutes, we awarded each rater a bonus to raise their pay to an average of $10 per hour. Each example was shown to three raters, and annotations were rejected if the rater gave a lower overall score to the random output than to the ground-truth one. A total of 3 annotations were rejected. Overall, the Fleiss' kappa agreement of pairs of annotators giving the same numerical score to the same question was 0.26.

Figure 5.2 shows the results. On the FitB task, the pre-trained and finetuned T5 models were indistinguishable in terms of quality. The Llm that formed continuations prompted with only the left context did somewhat better than the few-shot Llm, indicating that few-shot learning is not yet a feasible alternative to finetuning. On the continuation task, the Llm has the highest rating, which is unsurprising since it is a much larger model than T5. However, the finetuned T5 is rated almost as highly. Overall, these results suggest that T5, unlike the Llm, can be used effectively for continuation as well as FitB. Furthermore, if one doesn't care about controllability, T5 can be used effectively for both tasks without any finetuning.

Benefits of Controllability

There are good reasons to care about controllability. For example, length conditioning is extremely important for FITBmodels, since it is not possible to control the generation length by simply sampling more or fewer tokens. Pre-trained T5 tends to produce infill proposals which are shorter than the groundtruth (Figure A**??**), and there is no way to ask the model to produce longer generations. In contrast, finetuned T5 was able to produce generations in the target length bucket over 74% of the time (Table 5.4). Goal conditioning, while not strictly necessary for either either task, has been shown to be useful for generative commonsense reasoning [101] and may empower users in downstream applications such as AI-assisted creative writing [140]. Finetuned T5 is able to use all of the specified goal words over 79% of the time.

Domain Transfer

Prior work on FITB tends to only evaluate models trained on data from the same domain as the validation set. Our results show that despite training exclusively on C4, T5 models have strong transferability to more targeted domains such as Reddit Writing Prompts. This sort of transferability is extremely important for achieving the goal of having single models which can handle many tasks and domains.

5.3.5 Conclusion

In this work, we make the case for starting with a model capable of filling in the blank when attempting to build a system that can perform both FITB and continuation. As LMs become bigger, it will be unsustainable to have separately trained models per task. Multi-task, domain-transferable models with like the ones we propose require less total training and are more efficient to store and use at inference time. While pre-trained T5 by itself is capable of both infilling and continuation, additional conditioning signals such as desired length and goal text can be successfully incorporated into fine-tuning in order to support an even greater diversity of model interactions. Finally, we present a negative result that while few-shot learning is a promising method for building multi-task support without any finetuning, it is challenging to make work for the FitB task.

5.3.6 Summary of Contributions

The work described in this section was published in the 2022 Proceedings of the North American Association of Computational Linguistics **{TODO: citation}**. The work was performed with my collaborators Daphne Ippolito, Liam Dugan, Emily Reif, Ann Yuan, Andy Coenen, and Chris Callison-Burch. I led this project, designed and ran all experiments, and performed most of the analysis.

5.4 SUPPORTING ARBITRARY STYLE TRANSFER

?? Text style transfer is the task of rewriting text to incorporate additional or alternative stylistic elements while preserving the overall semantics and structure. Early approaches to style transfer required *parallel* text data [196, 135], where every input in the source style has a corresponding output in the target style. Because the availability of such data is limited, however, there has been a shift toward approaches which instead rely on *non-parallel* monostyle data [100, 79, 105, 87]. Most recently, *label-free* methods have taken advantage of the natural manifold of language to train style transfer models that require only a few exemplars in the target style for inference [182, 138]. This is true even for approaches which claim to be label-free [182, 138]. Hence, there is a clear need for new methods that both reduce the training data requirements and expand the scope of styles supported [78, 70].

In this section, we present *augmented zero-shot learning*, a prompting method that allows large language models to perform text style transfer to arbitrary styles, without any exemplars in the target style. Our method builds on prior work showing that sufficiently large LMs such as GPT-3 can perform various tasks ranging from classification to translation, simply by choosing a clever prompt to prepend to the input text for which the model is asked to continue [22, 17]. Although large LMs are trained only for continuation, recent work has shown that they can perform a variety of NLP tasks by expressing the task as a prompt that encourages the model to output the desired answer as the continuation [106, 129, 177, 22, 143]. The simplest approach, **zero-shot prompting**, directly uses natural language to ask the large LM to perform a task, as shown in Figure 5.4a. Zero-shot learning, however, can be prone to failure modes such as not returning well-formatted or logical outputs (see §5.4.4). However, zero-shot prompts are prone to failure modes such as not returning a well formatted or logical answer. This problem can often be overcome by prepending exemplars to the prompt that demonstrate what successful completions may look like. This approach, called **few-shot prompting**, has been shown to achieve higher performance, but requires exemplars for the exact task that we want the model to perform (Figure 5.4b).

To remove the need for these labeled exemplars for each style transfer task, we propose *augmented zero-shot learning*, a method for performing multi-task style transfer using a single set of exemplars. Instead of prompting the model with exemplars specific to the exact style transfer task we wish to perform, we prompt the model with examples of a variety of sentence rewriting operations, as shown in Figure 5.4c. This intuition is inspired by Reynolds and McDonell [137]'s observation that successful prompts constrain the behavior of the large LM away from failure modes—in our case, we aim to preserve the flexibility of a zero shot prompt while encouraging the model to produce outputs of a specific template. We keep the the format of the exemplars constant and insert the desired sentence transformation into the same format. In this way, the augmented zero-shot formulation supports arbitrary sentence rewriting tasks without the need to write any task-specific exemplars. Thus, it works for a wide range of styles, including modifying the text to be "*more melodramatic,*" "*insert a metaphor,*" or "*include the word balloon.*"

Augmented zero-shot learning is simple and facilitates the application of style transfer to a wider range of styles than existing work. We also foresee that our strategy of prompting an LLM with non-task specific examples will prove useful for other tasks besides style transfer.

Table 5.6: Example style transfer outputs from augmented zero-shot learning for non-standard styles.

Original	`There, in the middle of the street, stood an old man in a weatherbeaten brown coat.`
More descriptive	`The man's brown trenchcoat was coated with snow, and as he leaned against the telephone booth, he cast a long shadow across the road in the late afternoon sun.`
Include a metaphor	`There, in the middle of the street, stood an old man whose face resembled the weatherworn bark of a gnarled tree.`
Include the word "*park*"	`There, in the middle of Central Park, stood an old man in a weatherbeaten brown coat.`
Include the word "*balloon*"	`There, in the middle of the street, stood an old man with several colourful balloons tied to the straps of his coat.`
More comic	`There, in the middle of the street, stood an old man in a raggedy brown coat that was much too big for him.`
More melodramatic	`There, in the middle of the road, stood a grizzled old man, the light of life faded from his sunken eyes.`

5.4.1 Experimental Setup

Style Transfer Tasks

We consider six style transfer tasks that we deem non-standard, listed in Table 5.6. These styles were chosen to be representative of most frequent style adjustments made by users of an AI-assisted text editor that employs our method (discussed further in §5.4.3). As source sentences, we use 50 sentences randomly drawn from the Reddit Writing Prompts validation set [49], excluding those that already clearly exhibited one of the styles or were ungrammatical/incoherent. We use human evaluation for these styles, since not all styles have readily available classifiers.

We also evaluate our method on two standard style transfer tasks: sentiment and formality. We use the Yelp polarity dataset [191] for sentiment and Grammarly's Yahoo Answers Formality Corpus (GYAFC) dataset for formality [135].[20] These datasets allow us to evaluate performance of augmented zero-shot learning in the context of prior supervised methods which have been used on these tasks.

20 Hosted by Luo et al. [108].

Model

?? Augmented zero-shot learning requires a large language model. We primarily use LaMDA, a left-to-right decoder-only transformer language model [166] with a non-embedding parameter count of 137B [161]. The pre-trained LaMDA model, which we refer to as *LLM*, was trained on a corpus comprising 1.95B public web documents, including forum and dialog data and Wikipedia. The dataset was tokenized into 2.49T BPE tokens with a SentencePiece vocabulary size of 32K [89]. We also use *LLM-Dialog*, the final LaMDA model which was finetuned on a curated, high-quality subset of data identified to be in a conversational format. Decoding was done with top-k=40. To show that the success of augmented zero-shot learning is not restricted to these two large LMs, we also perform experiments with GPT-3 (Table 5.8). For GPT-3, decoding was done with nucleus sampling using p=0.6 [69].

The full prompts used for *LLM* and GPT-3 are shown in Figure 5.12. For *LLM-Dialog*, the prompt was instead formulated as a conversation between one agent who is requesting rewrites and another who is performing the rewrites.

5.4.2 Results

Non-Standard Styles

For our six non-standard styles, we asked six professional raters to assess <input sentence, target style, output sentence> tuples. These raters are fluent in English, live in India, and work full time labeling and evaluating data. To decrease inter-rater discrepancy and ensure that our instructions were clear, we had an initial calibration session where they test-rated a small portion of the data (around 10 datapoints which were then omitted from the results) and asked us any clarifying questions. For each style, we compare outputs from our method plus the three baselines for 50 sentences.

Each tuple was scored by three raters (3,600 ratings total) on the following three axes which are standard to textual style transfer [116]: **(1) transfer strength** (the amount that the output actually matches the target style),

(2) semantic preservation (whether the underlying meaning of the output text, aside from style, matches that of the input), and **(3) fluency** (whether the text is coherent and could have been written by a proficient English speaker). Following Sakaguchi and Van Durme [141], transfer strength and semantic preservation were rated on a scale from 1–100. A screenshot of the evaluation UI is shown in Figure 5.5. We use *dialog-LLM*, and compare it with three other methods: **(1) zero-shot** (a baseline), **(2) paraphrase** (our normal augmented zero shot prompt, but with the target style of *"paraphrased"*, as a control) and **(3) human** (ground-truth transformations written by the authors).

Figure 5.6 shows these results. We found that the outputs of our method were rated almost as highly as the human-written ground truth for all three evaluations. The zero-shot baseline performed the worst in all categories: 25.4% of the time, it did not return a valid response at all (see §5.4.4), compared with 0.6% for augmented zero shot. The strong performance of the paraphrase baseline at fluency and semantic similarity shows that large LMs are capable of generating high quality text that remains true to the input sentence's meaning. Overall, the average length of the input sentences was 66 characters, whereas the average length of augmented zero-shot outputs was 107 characters. For context, human paraphrase outputs were 82 characters.

For a subset of the tasks, some automatic evaluation was also possible. We found that the "*balloon*" and "*park*" transformations successfully inserted the target word 85% of the time. For "*more descriptive*" and "*include a metaphor*" the transformed text was, as expected, longer than the original (by 252% and 146% respectively, compared with 165% and 146% for human baselines).

Standard Styles

To better contextualize the performance of our method with prior methods, we also generated outputs for two standard style transfer tasks: sentiment and formality. Figure 5.7 shows human evaluations (same setup as before) for our outputs as well as the outputs from two popular prior style transfer methods, Unsup MT [128] and Dual RL [108]. The outputs from our method were rated comparably to both human generated responses and the two prior methods, using the same rating setup as the non-standard styles, with six outputs and baselines for four styles across 50 sentences, rated independently by three raters, totalling 3,000 total ratings.

Furthermore, following Li et al. [100] and Sudhakar et al. [155], we perform automatic evaluation for sentiment style transfer since there are classifiers available for these styles. We note that although automatic evaluations can diverge from human ratings, they can still be a good proxy as we could not perform human evaluation against every prior method due to time and resource constraints. We automatically evaluate **(1) transfer strength** using a sentiment classifier from HuggingFace Transformers [178], **(2) semantic similarity** to human examples provided by Luo et al. [108] via BLEU score, and **(3) fluency** via perplexity, as measured by GPT-2 (117M).

Table 5.7 shows these automatic evaluations, with four main takeaways. First, augmented zero-shot prompting achieves high accuracy and low perplexity compared with baselines. The BLEU scores, however, the outputs of our model had low BLEU scores with respect to human generated outputs 5.7. Based on qualitative examination of outputs, we believe that this is because our model outputs often used different language from human annotations, despite having high semantic similarity with the source sentence. For instance, for transferring the sentiment of "*ever since joes has changed hands it's just gotten worse and worse*" to positive sentiment, our augmented zero-shot learning model outputted "the establishment has continued to provide excellent service, improving steadily since its change of ownership." This will have low BLEU with the ground truth with respect to human references, which is simply "*ever since joes has changed hands it's just gotten better and better*." Though we do not see this as an inherent problem, increasing the BLEU for the purposes of comparison can be done in an easy way via candidate selection, as our model returns sixteen possible continuations. In applications for which we prefer model outputs to have high lexical similarity to the source sentence, we could select the candidate of the sixteen with the highest BLEU score compared with the original source sentence. We find that this candidate selection step can substantially improve the BLEU score with the ground truth target sentences, as we show in Table 5.8.

Second, we apply augmented zero-shot learning to GPT-3 175B; these results indicate that augmented zero-shot learning generalizes to another large language model. Third, we vary model size for GPT-3 models, finding

Table 5.7: Comparing augmented zero-shot prompting with supervised style transfer methods on the Yelp sentiment style transfer dataset using automatic evaluation. Acc: accuracy; PPL: perplexity. The inference-only table shows our method applied to 3 different sizes of GPT-3, plus our own LLM.

	Acc	BLEU	PPL
SUPERVISED METHODS			
Cross-alignment [148]	73.4	17.6	812
Backtrans [128]	90.5	5.1	424
Multidecoder [53]	50.3	27.7	1,703
Delete-only [100]	81.4	28.6	606
Delete-retrieve [100]	86.2	31.1	948
Unpaired RL [181]	52.2	37.2	2,750
Dual RL [108]	85.9	55.1	982
Style transformer [38]	82.1	55.2	935
INFERENCE-ONLY METHODS			
GPT-3 ada, aug zero-shot	31.5	39.0	283
GPT-3 curie, aug zero-shot	53.0	48.3	207
GPT-3 da vinci, aug zero-shot	74.1	43.8	231
LLM: zero-shot	69.7	28.6	397
five-shot	83.2	19.8	240
aug zero-shot	79.6	16.1	173
LLM-dialog: zero-shot	59.1	17.6	138
five-shot	94.3	13.6	126
aug zero-shot	90.6	10.4	79

that larger size greatly improves style transfer. Fourth, for *LLM* and *LLM-dialog*, we find that augmented zero-shot learning substantially outperforms vanilla zero-shot learning and almost reaches the accuracy of five-shot learning.

Comparison with a range of prior methods

To compare against a larger range of prior supervised methods, we used automatic evaluation, and found comparable performance with the highest-scoring method for transfer strength. The results are shown in Table **??**. We were also significantly more fluent than all other methods. Finally, our method fell short on semantic

Table 5.8: Sentiment style transfer results with candidate selection (cand. select.). Candidate selection means that of the sixteen examples returned by our model, we choose the one with the highest BLEU with the source sentence.

	Acc	BLEU	PPL
LLM-128B			
Zero-shot	69.7	28.6	397
+ cand. select.	31.4	61.5	354
Five-shot	83.2	19.8	240
+ cand. select.	61.5	55.6	306
Augmented zero-shot	79.6	16.1	173
+ cand. select.	65.0	49.3	292
LLM-128B-dialog			
Zero-shot	59.1	17.6	138
+ cand. select.	46.8	24.2	166
Five-shot	94.3	13.6	126
+ cand. select.	81.3	47.6	345
Augmented zero-shot	90.6	10.4	79
+ cand. select.	73.7	40.6	184

preservation compared to other methods. However, BLEU is known to penalize long sentences, and the scores do not always align with human judgements. For example, our model's worse performance could be because it was not explicitly trained on Yelp data, so its generations are less likely to be in the style of Yelp reviews than models that were.

Comparison across Different LLMs

We also compared between three varieties of model: GPT-3 [22], LLM, and LLM-Dialog. We adjusted the prompt template slightly to accommodate these differences: for LLM and GPT-3, the prompt template replaced "`Rewrite it to be` **`<style>`**" with "`Here is a rewrite of the text, which is` **`<style>`**". For our augmented zero-shot prompts we also see that the LLM-dialog version had higher accuracy than the LLM and GPT-3, but lower BLEU. Based on qualitative inspection, we believe the lower BLEU is due to the LLM-dialog adding additional detail in the generated sentences, which is consistent with an "interestingness" objective that is typically encoded into dialog training.

Prompt Construction

Prompt engineering can be brittle: Reynolds and McDonell [137] describe how reformulating the language of a prompt can have significant impact on performance, and that finding the right prompt is for a task is more akin to locating an already-learned task than truly learning a new one. To explore this, we compared several variations of the prompts for sentiment, varying the language of the prompt to use "*more positive/negative*," "*happier/sadder*," "*more optimistic/pessimistic*," or "*more cheerful/miserable*." As shown in Table 5.9, performance differed across the four prompts, but we found them comparable. In a real world setting, our augmented zero-shot approach allows users to effortlessly try out many different phrasings for the task until they find one that performs satisfactorily.

Reynolds and McDonell [137] further emphasize that prompt engineering is mostly about avoiding various failure cases. In this work, we use delimiters ("{" and "}") to help avoid parsing errors, giving scores of zero when there was no valid responses with such delimiters. There are other delimiters that could be used (e.g.,

Table 5.9: Comparing variations of augmented zero-shot learning prompt wording for sentiment style transfer.

Model / prompt wording	Acc	Bleu	PPL
LLM			
"more positive/negative"	76.3	14.8	180
"happier/sadder"	62.6	15.5	173
"more optimistic/pessimistic"	69.7	14.1	143
"more cheerful/miserable"	74.5	15.7	186
LLM-Dialog			
"more positive/negative"	90.5	10.4	79
"happier/sadder"	85.9	9.6	90
"more optimistic/pessimistic"	85.8	10.2	79
"more cheerful/miserable"	88.8	11.4	93

Table 5.10: Requests in the form of "*Rewrite this...*" made by real users to a large LM-powered text editor.

into paragraphs • to be a bit clearer • to be a little less angsty • to be a word for a song • to be about mining • to be about vegetables • to be better written • to be less descriptive • to be less diabolical • to be more absurd • to be more adventurous • to be more angry • to be more cheerful • to be more descriptive • to be more Dickensian • to be more emotional • to be more fancy • to be more flowery • to be more interesting • to be more joyful • to be more magical • to be more melodramatic • to be more philosophical • to be more revolutionary • to be more scary • to be more subtle • to be more surprising • to be more suspenseful • to be more technical • to be more violent • to be more whimsical • to be warmer • to fit better grammatically with the rest of the story • to make more sense • to use a more interesting word • with a few words

quotes, "(" and ")", "<" and ">", newlines with a colon (as used by GPT-3), etc. We chose curly braces as they were 1) likely to occur in the training data as delimiters in other contexts and 2) not frequently part of the input sentence itself. We also use a second person prompt template for the dialog, which yielded better results as it was more similar to the training data. Exploring these options more quantitatively would be an interesting direction for future work.

5.4.3 Potential of Arbitrary Styles

One promising application of augmented zero-shot learning is an AI-powered writing assistant that can allow writers to transform their text in arbitrary ways that the writer defines and controls. As a qualitative case study to explore what arbitrary re-write styles may be requested, we built an AI-assisted story-writing editor with a "rewrite as" feature that uses our augmented few-shot method. Our editor has a freeform text box for users to specify how they would like a selection of their story to be rewritten (Figure 5.8). We asked 30 people from a creative writing group to use our UI to write a 100-300 word story, collecting 333 rewrite requests in total. Table 5.10 shows a list of unique rewrite requests collected from 30 early users recruited from a creative writing mailing list. These were as diverse as asking for the text "*to be about mining*" or "*to be less diabolical*." Overall, we collected 333 rewrite requests, of which users chose to insert the model's response into their story 18 (5.4%) times.

5.4.4 Limitations and Failure Modes

This section details several qualitative limitations with our method.

Unparsable answers

A frequent problem that arises when using large LMs for other NLP tasks is their outputs cannot be automatically parsed into usable answers. For example, when given a prompt like `"Here is some text: that is an ugly dress. Here is a rewrite of the text, which is more positive"` *LLM-Dialog* might return something like `"Sounds like you are a great writer!"` Similar error modes exist for *LLM*, which might output something like `"Here are more writing tips and tricks."` Other times, the response contains correct information, but it cannot be automatically parsed (e.g., `"a good rewrite might be to say that the dress is pretty."`) In hindsight, these outputs make a lot of sense: most of the training data of large LMs is not well-formatted pairs of inputs and outputs [137]. See §5.4.2 for how we dealt with these issues.

Hallucinations

Large LMs are known to hallucinate text content; we saw this happen frequently for style transfer. While this is an advantage in some contexts like creative writing, it is undesirable for applications like summarization.

Inherent style trends

We also noticed that even our *"paraphrase"* baseline, where the model was simply asked to rewrite the input sentence, was rated highly for style strength for a few styles, including *"more formal"* and *"more melodramatic"*. This implies that our method's generations generally trend toward these styles. A direction for future work would be to see what styles and qualities of text our method (and large LMs in general) are inherently more likely to produce.

Less reliable than trained methods

For style transfer tasks that have available training data, prior methods that either train or finetune on that data are going to be inherently more reliable at producing text that looks like their training data. This can be observed in the lower BLEU scores our method achieves than trained methods, despite comparable transfer accuracy. Thus, augmented zero-shot learning offers less fine-grained controllability in the properties of the style-transferred text than methods which see task-specific training data.

Large LM safety concerns

Large LMs themselves come with their own host of difficulties, barriers to entry, and potential safety concerns as discussed by **bender2021stochastic**, which are also valid for this style transfer method. However, we also think that this method can be a useful tool in exploring and exposing the safety and boundaries of these models themselves: what happens if we try to force the large LM to make a text "more racist", "more sexist", or "more incendiary"? It is important to keep pushing these models to their boundaries to see where they fail and where problems arise, and specific use cases that show a broader range of the model's capabilities also show a broader range of its failure modes.

Table 5.11: Examples of users' arbitrary style transfer requests for which the model suggestion was accepted.

Input	**Model suggestion**
You honor me," said Bruno. As he spoke, he began to laugh. His laugh was merry sounding and melodic. He took a deep breath. When he spoke again, his voice was low and strong. "I take this great task on and with my brothers shall lead us on to victory! **Rewrite it to be about mining.**	You honor me, said Bruno. As he spoke, he began to laugh. His laugh was deep and gravely, as if from the bowels of the earth. He took a deep breath. When he spoke again, his voice was deep and dark.'I take this great task on and with my brothers shall delve into the mines!"
I reckon,' smiled Toe, 'you'd have to say the same for icicles, wouldn't you? **Rewrite it to be more interesting.**	'I reckon,' smiled Toe, 'you'd have to say the same for icicles, wouldn't you?' These words were barely out of his mouth before the others realized exactly what they meant.
with its southern perfume **Rewrite it to be more descriptive**	A mixture of sweet honeysuckles and jasmine with a hint of tobacco and warm mesquite smoke. It was his southern perfume, with a hint of a memory of simpler times.
But Sally remained unmoved–and lonely. **Rewrite it to be more emotional**	Her tears rolled and rolled down her cheek, mixing with the saltwater around her, until the water was a rich mixture of the two. She wanted her poem to be cheerful like it used to be, but couldn't find the right words.

5.4.5 Summary of Contributions

The work described in this section was published in the 2022 Proceedings of the Association of Computational Linguistics **{TODO: citation}**. The work was performed with my collaborators **{TODO: author names}**. I worked with Emily Reif to come up with the premise for the project: of formulating style transfer as an

Table 5.12: In black, we show the exact augmented-zero shot prompts used in our experiments, for *LLM* and GPT-3 (top), and for *LLM-Dialog* (bottom). As shown, for *LLM-Dialog*, we replaced "`Here is a rewrite of the text, which is`" with "`Rewrite it to be`". Each line starting with ">"" above was passed in as an individual dialog turn. The blue shows how an input text and goal style are concatenated to the few-shot prompt in order to produce final model output. Note that we can achieve high accuracy even though the prompt formulation resulted in some minor grammatical errors for some styles (e.g., "`rewrite it to be include the word 'snow'`"). Text versions of these prompts can be downloaded at `https://bit.ly/3fLDuci`.

Augmented Zero-shot Prompt: LLM

```
Here is some text: {When the doctor asked Linda to take the medicine, he smiled and gave her a lollipop.}. Here is a rewrite of the text,
which is more scary. {When the doctor told Linda to take the medicine, there had been a malicious gleam in her eye that Linda didn't
like at all.} Here is some text: {they asked loudly, over the sound of the train.}. Here is a rewrite of the text, which is more intense.
{they yelled aggressively, over the clanging of the train.} Here is some text: {When Mohammed left the theatre, it was already dark out}.
Here is a rewrite of the text, which is more about the movie itself. {The movie was longer than Mohammed had expected, and despite the
excellent ratings he was a bit disappointed when he left the theatre.} Here is some text: {next to the path}. Here is a rewrite of the
text, which is about France. {next to la Siene} Here is some text: {The man stood outside the grocery store, ringing the bell.}. Here is
a rewrite of the text, which is about clowns. {The man stood outside the circus, holding a bunch of balloons.} Here is some text: {the
bell ringing}. Here is a rewrite of the text, which is more flowery. {the peales of the jangling bell} Here is some text: {against the
tree}. Here is a rewrite of the text, which is include the word "snow". {against the snow-covered bark of the tree}     Here is some text:
{That is an ugly dress}. Here is a rewrite of the text, which is more positive."
```

Augmented Zero-shot Prompt: LLM-dialog

```
> Here is some text: {When the doctor asked Linda to take the medicine, he smiled and gave her a lollipop.}. Rewrite it to be more scary.
> {When the doctor told Linda to take the medicine, there had been a malicious gleam in her eye that Linda didn't like at all.}
> Here is some text: {they asked loudly, over the sound of the train.}. Rewrite it to be more intense.
> {they yelled aggressively, over the clanging of the train.}
> Here is some text: {When Mohammed left the theatre, it was already dark out}. Rewrite it to be more about the movie itself.
> {The movie was longer than Mohammed had expected, and despite the excellent ratings he was a bit disappointed when he left the
theatre.}
> Here is some text: {next to the path}. Rewrite it to be about France.
> {next to la Siene}
> Here is some text: {The man stood outside the grocery store, ringing the bell.}. Rewrite it to be about clowns.
> {The man stood outside the circus, holding a bunch of balloons.}
> Here is some text: {the bell ringing}. Rewrite it to be more flowery.
> {the peals of the jangling bell}
> Here is some text: {against the tree}. Rewrite it to be include the word "snow".
> {against the snow-covered bark of the tree}
> Here is some text: {That is an ugly dress}. Rewrite it to be more positive."
```

arbitrary rewriting operation. I helped to design the experiments and contributed significantly to analysis of the results.

5.5 WORDCRAFT: AN EDITOR FOR AI-ASSISTED WRITING.

How do we evaluate natural language generation in realistic settings? In all the human evaluations discused so far in my thesis, we recruited annotators to evaluate text in artifical settings. In Sections 3.4 and 3.5, university students were asked to label text as human-written or machine-generated in user interfaces custom-tailored to collect these sorts of annotations. In Sections 5.3 and Section **??**, novel natural language systems were evaluated by asking raters on Amazon Mechnical Turk to view and score generated text. Notably, the annotators were not asked to use the proposed NLG tools to accomplish a task of interest; they were only asked to evaluate pre-generated outputs.

When building novel systems for cotrollable text generation, it is crucial to keep in mind why these tools are being built and who their target audience might be. This is especially true in the area of AI-assisted creative writing, where there is often significant discrepency between how novel tools are evaluated (Amazon Mechanical Turker workers paid per annotation) and how they are intended to be used in the real world (writers seeking support or inspiration while performing their craft).

Wordcraft is an AI-augmented text processor which is intended as a real-world test bed for controllable text generation paradigms in the domain of creative writing. Our goal in developing Wordcraft was to learn how people interact with modern state-of-the-art NLG systems, what tasks they ask the NLG systems to do do and how well the systems deliver, and how this feeds back into the works people ultimately create. The user interface for Wordcraft consists of of a traditional text editor alongside a set of controls that vary based on where the user's cursor is and whether they have selected any text. The user also has access to a chatbot they can talk about their story with.

In this section, I describe the controls included in Wordcraft, how they're implemented, and the motivation for including them. I then describe how Wordcraft offers the chance for a more realistic evaluation of state-of-the-art language generation systems through user studies with both amateur and expert writers.

5.5.1 Features of Wordcraft

Wordcraft uses few-shot learning techniques [23] to support a variety of generative controls. The underlying neural language model backing all these interactions is LaMDA [161], which is described in detail in Section **??**. Each control shows several candidate generations, which the user can select between. When a generation is chosen, it will be inserted into the text at the location of the user's cursor.

Continuation

As described in Section **??**, continuation is the default action of a left-to-right neural language model such as LaMDA, achieved simply by prompting the LM with a text passage, and decoding a possible continuation. A continuation generator is useful for writers who want text appended to the end of what they have written so far, and thus this control is only available when the user's cursor is at the end of the text passage.

Story Ideas

When the user has no text inputted, a few-shot learning prompt is used to suggest story ideas. The following prompt is used:

```
Foo
```

Fill in the blank

Elaboration

Custom Prompting

5.5.2 User Study

5.5.3 Summary of Contributions

The Wordcraft AI-assited writing tool was first published in the HCI+NLP Workshop colocated with the 2021 Meetings of the European Association of Computation Linguistics [35]. The user study was subsequently published at the 2022 International Conference on Intelligent User Interfaces [186]. Wordcraft was built alongside by collaborators Ann Yuan, Emily Reif, and Andy Coenen,. In particular, Ann, Emily, and Andy did almost all the software engineering for Wordcraft, and Ann Yuan led the user study. I worked on designing features for the user interface and proposing and prototyping various of the controls described in Section 5.5.1. I also assisted with designing the user study and analyzing its results.

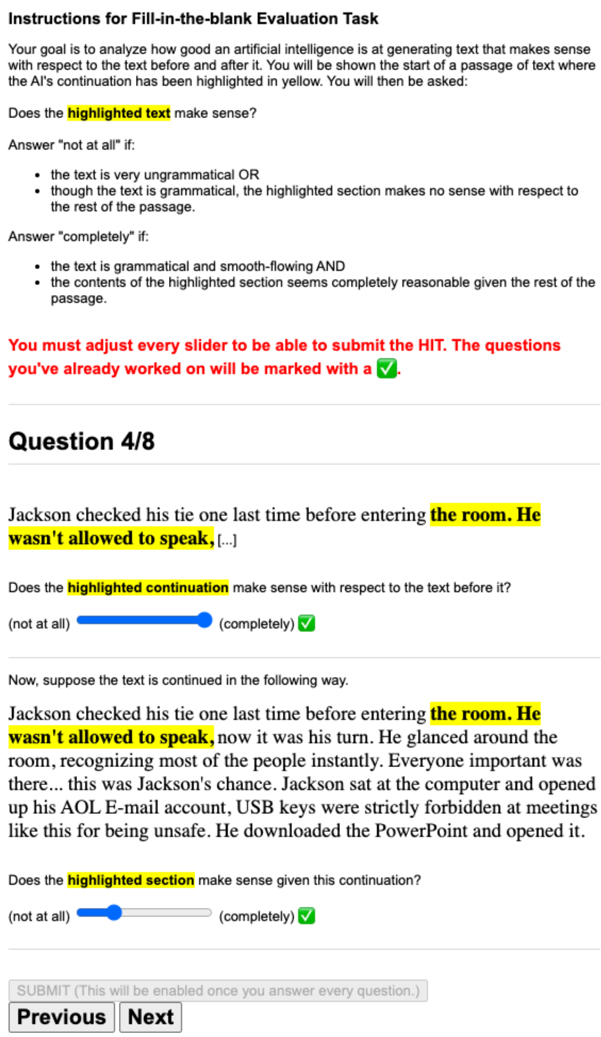


Figure 5.3: A screenshot of the question structure for human evaluation.

Zero-shot learning prompt

Here is some text: {That is an ugly dress}. Here is a rewrite of the text, which is *more positive*: {

(a)

Few-shot learning prompt

Here is some text: {I was really sad about the loss}. Here is a rewrite of the text, which is ***more positive***: {I was able to accept and work through the loss to move on.}

Here is some text: {The eggnog was tasteless}. Here is a rewrite of the text, which is ***more positive***: {The eggnog had a great, festive taste to it.}

…

Here is some text: {That is an ugly dress}. Here is a rewrite of the text, which is *more positive*: {

(b)

Augmented zero-shot learning prompt (ours)

Here is some text: {When the doctor asked Linda to take the medicine, he smiled and gave her a lollipop}. Here is a rewrite of the text, which is ***more scary***: {When the doctor told Linda to take the medicine, there had been a malicious gleam in her eye that Linda didn't like at all}

Here is some text: {They asked loudly, over the sound of the train}. Here is a rewrite of the text, which is ***more intense***: {They yelled aggressively, over the clanging of the train}

…

Here is some text: {That is an ugly dress}. Here is a rewrite of the text, which is *more positive*: {

more melodramatic | ***includes a metaphor*** | ***include the word "balloon"***

(c)

Figure 5.4: Zero-shot, few-shot, and augmented zero-shot prompts for style transfer. The boldface text is the zero-shot prompt, and the plain text is the additional priming sequence. The full prompts used in this paper are shown in Table 5.12. We encourage readers to examine the outputs of our model at `https://bit.ly/3fLDuci`.

Instructions: In this task, your goal is to identify whether a desired transformation has been successfully applied to a sentence, without changing the overall meaning of the sentence. Each question contains a sentence marked "original sentence," a desired transformation, and an output sentence where the transformation has been applied.

Each of these questions relates to the same original text and desired transform, but each has a different output transformed sentence. Please rate each transformed sentence along the following three axes:

1) Transferred Style Strength: Does the transformed text has the applied style/transform compared to the original text? For example, if the original text is "I went to the store" and the style is "more angry":

example	score	reasoning
"The store is where I went"	0	The transformed text is no more angry than the original text.
"I went to the stupid store"	50	The transformed text somewhat relates to the style.
"When I went to the store, I couldn't believe how rude the storekeeper was to me!"	100	The text is clearly more angry.

2) Meaning: Does the transformed sentence still have the same overall meaning as the original? It is OK if extra information is added, as long as it doesn't change the underlying people, events, and objects described in the sentence. You should also not penalize for meaning transformations which are necessary for the specified transformation. For example, if the original text is "I love this store" and the style is "more angry":

example	score	reasoning
"it is raining today"	0	the transformed text is about something totally different. It would be hard to tell that the texts are related at all.
"they were out of chicken at the store"	50	The transformed text is mostly related to original-- some modifications of the meaning have been made but they are not egregious
"I adore the store." or "The store was really horrible; it took forever to do my shopping."	100	The text talks about the same concepts as the original, just with different or more words

3) Fluency: Is this sentence fluent english and does it make sense?

example	score	reasoning
"who said that? I thought we were going to go together!"	Yes	This text makes sense
"who, she said it up to me and to me together!"	No	The text is incoherent

Original text: "Everyone in my world had different eye colours."

Desired transformation: more melodramatic

Transformed text: "Everyone in my world had the most intensely colorful eyes, and no one in this world can possibly understand how beautiful they were."

1) Transferred Style Strength: The transformed text has the applied style/transform.

50

2) Meaning: The meaning is preserved between the original and transformed texts (ignoring the ways that the style/transform would change the meaning)

50

3) Fluency: the transformed text is fluent English and it makes sense.

- Yes
- No

Figure 5.5: The rating UI used for human evaluation. The user may be shown a number of blue squares at once with the same original text and different outputs.

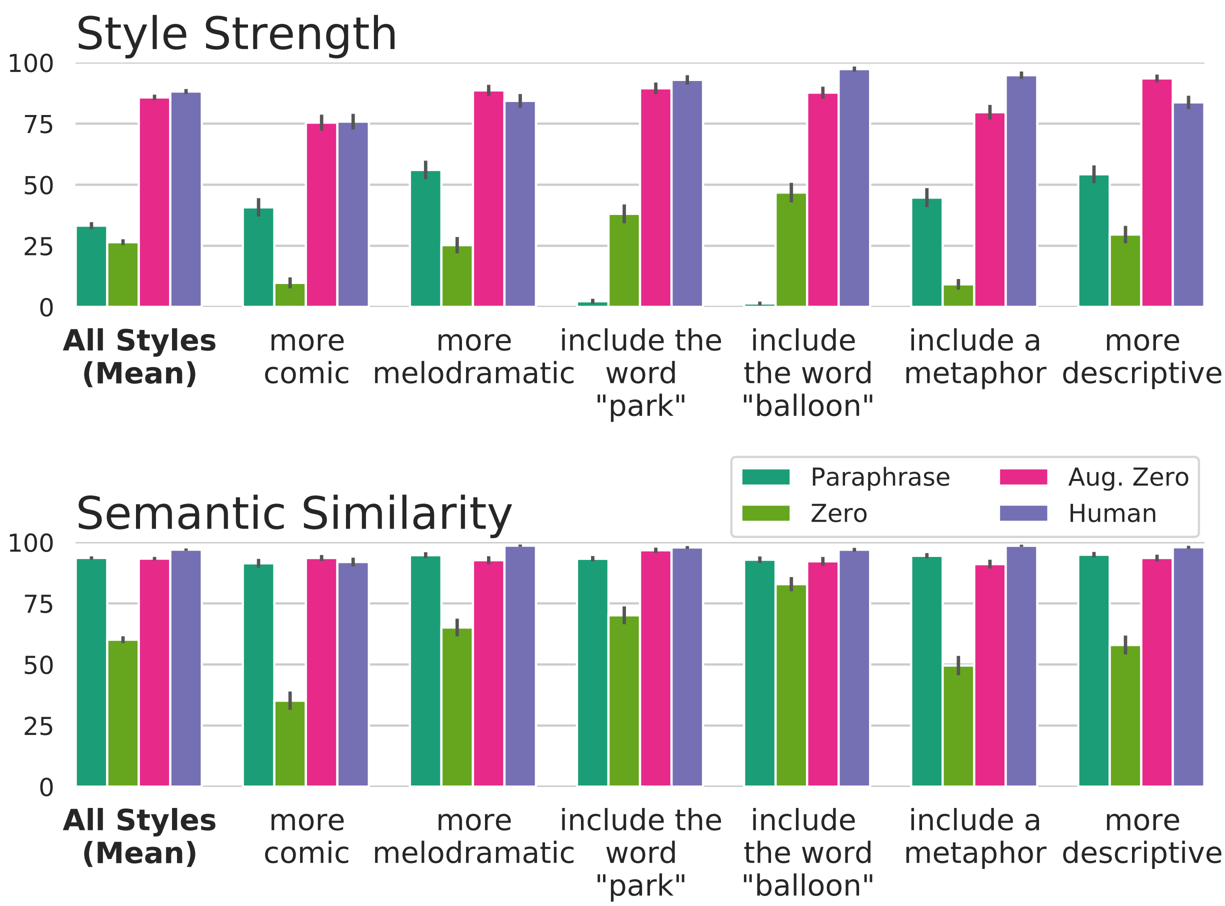


Figure 5.6: Human evaluation of style transfer for six atypical styles. Our method is rated comparably to the human-written ground truth. Error bars are mean standard error.

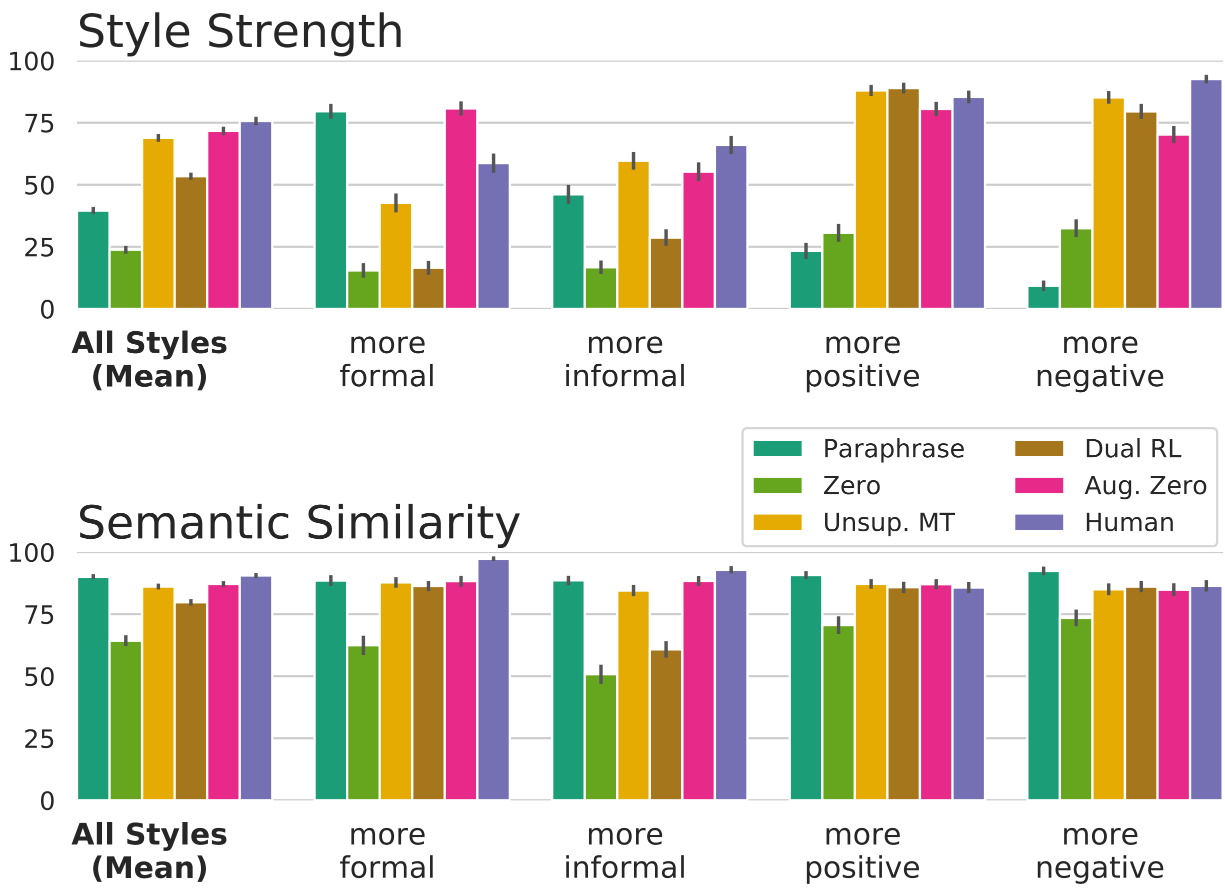


Figure 5.7: Human evaluation of sentiment and formality transfer. Our method is rated comparably to human-written ground truth as well as prior methods. Error bars show Standard Error of the Mean. Unsup. MT is Prabhumoye et al. [128]; Dual RL is Luo et al. [108].

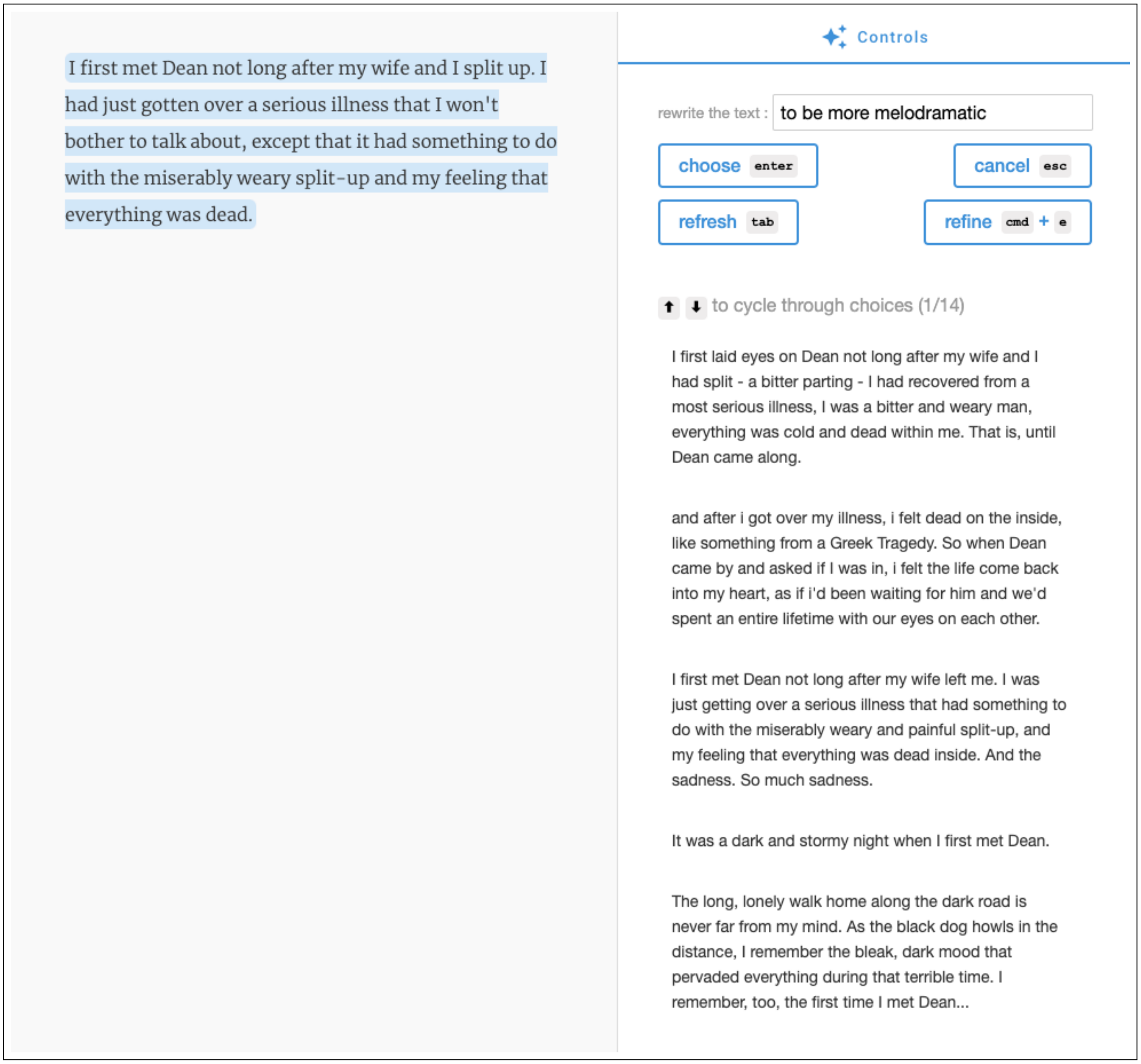


Figure 5.8: Screenshot AI-assisted editor with 'Rewrite as' feature.

6 CONCLUSIONS

I conclude things.

LIST OF TABLES

LIST OF ILLUSTRATIONS

BIBLIOGRAPHY

[1] Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. "Deep learning with Differential Privacy." In: *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*. 2016, pp. 308–318.

[2] David Ifeoluwa Adelani, Haotian Mai, Fuming Fang, Huy H Nguyen, Junichi Yamagishi, and Isao Echizen. "Generating sentiment-preserving fake online reviews using neural language models and their human-and machine-based detection." In: *International Conference on Advanced Information Networking and Applications*. Springer. 2020, pp. 1341–1354.

[3] Nader Akoury, Shufan Wang, Josh Whiting, Stephen Hood, Nanyun Peng, and Mohit Iyyer. "STORIUM: A Dataset and Evaluation Platform for Machine-in-the-Loop Story Generation." In: *arXiv preprint arXiv:2010.01717* (2020).

[4] Miltiadis Allamanis. "The adverse effects of code duplication in machine learning models of code." In: *Proceedings of the 2019 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software*. 2019, pp. 143–153.

[5] Hunt Allcott and Matthew Gentzkow. "Social media and fake news in the 2016 election." In: *Journal of economic perspectives* 31.2 (2017), pp. 211–36.

[6] Peter Anderson, Basura Fernando, Mark Johnson, and Stephen Gould. "SPICE: Semantic Propositional Image Caption Evaluation." In: *European Conference on Computer Vision*. 2016. URL: https://arxiv.org/abs/1607.08822.

[7] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. "Bottom-up and top-down attention for image captioning and visual question answering." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6077–6086. URL: http://openaccess.thecvf.com/content_cvpr_2018/CameraReady/1163.pdf.

[8] Rohan Anil, Badih Ghazi, Vineet Gupta, Ravi Kumar, and Pasin Manurangsi. "Large-scale differentially private BERT." In: *arXiv preprint arXiv:2108.01624* (2021).

[9] Devansh Arpit, Stanisław Jastrzębski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, et al. "A closer look at memorization in deep networks." In: *International Conference on Machine Learning*. PMLR. 2017, pp. 233–242.

[10] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. "Program synthesis with large language models." In: *arXiv preprint arXiv:2108.07732* (2021).

[11] Ashutosh Baheti, Alan Ritter, Jiwei Li, and Bill Dolan. "Generating More Interesting Responses in Neural Conversation Models with Distributional Constraints." In: *Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*. 2018.

[12] Anton Bakhtin, Sam Gross, Myle Ott, Yuntian Deng, Marc'Aurelio Ranzato, and Arthur Szlam. "Real or Fake? Learning to Discriminate Machine from Human Generated Text." In: *arXiv preprint arXiv:1906.03351* (2019).

[13] Jack Bandy and Nicholas Vincent. *Addressing "Documentation Debt" in Machine Learning Research: A Retrospective Datasheet for BookCorpus*. 2021. arXiv: 2105.05241 [cs.CL].

[14] Emily M. Bender and Batya Friedman. "Data Statements for Natural Language Processing: Toward Mitigating System Bias and Enabling Better Science." In: *Transactions of the Association for Computational Linguistics* 6 (2018), pp. 587–604. DOI: 10.1162/tacl_a_00041. URL: https://www.aclweb.org/anthology/Q18-1041.

[15] Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. *GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow*. Version 1.0. If you use this software, please cite it using these metadata. Mar. 2021. DOI: 10.5281/zenodo.5297715. URL: https://doi.org/10.5281/zenodo.5297715.

[16] Burton H Bloom. "Space/time trade-offs in hash coding with allowable errors." In: *Communications of the ACM* 13.7 (1970), pp. 422–426.

[17] Gwern Branwen. "GPT-3 Creative Fiction." In: (2020). URL: https://www.gwern.net/GPT-3.

[18] Andrei Z Broder. "On the resemblance and containment of documents." In: *Proceedings. Compression and Complexity of SEQUENCES 1997 (Cat. No. 97TB100171)*. IEEE. 1997, pp. 21–29.

[19] D. W. Brown. *Corpus of Presidential Speeches*. Retrieved from http://www.thegrammarlab.com. 2016.

[20] Hannah Brown, Katherine Lee, Fatemehsadat Mireshghallah, Reza Shokri, and Florian Tramèr. *What Does it Mean for a Language Model to Preserve Privacy?* 2022. arXiv: 2202.05520 [stat.ML].

[21] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[22] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. "Language Models are Few-Shot Learners." In: *CoRR* abs/2005.14165 (2020). arXiv: 2005.14165. URL: https://arxiv.org/abs/2005.14165.

[23] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. "Language models are few-shot learners." In: *Advances in Neural Information Processing Systems 33*. 2020.

[24] Daniel Buschek, Martin Zürn, and Malin Eiband. "The impact of multiple parallel phrase suggestions on email input and composition behaviour of native and non-native english writers." In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 2021, pp. 1–13.

[25] Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramer, and Chiyuan Zhang. "Quantifying Memorization across Neural Language Models." In: *arXiv preprint arXiv:2202.07646* (2022).

[26] Nicholas Carlini, Chang Liu, Úlfar Erlingsson, Jernej Kos, and Dawn Song. "The secret sharer: Evaluating and testing unintended memorization in neural networks." In: *USENIX Security Symposium*. 2019.

[27] Nicholas Carlini, Florian Tramer, Eric Wallace, Matthew Jagielski, Ariel Herbert-Voss, Katherine Lee, Adam Roberts, Tom Brown, Dawn Song, Ulfar Erlingsson, Alina Oprea, and Colin Raffel. *Extracting Training Data from Large Language Models*. 2020. arXiv: `2012.07805 [cs.CR]`.

[28] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. "One billion word benchmark for measuring progress in statistical language modeling." In: *arXiv preprint arXiv:1312.3005* (2013).

[29] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. "Evaluating large language models trained on code." In: *arXiv preprint arXiv:2107.03374* (2021).

[30] Hung Chim and Xiaotie Deng. "A New Suffix Tree Similarity Measure for Document Clustering." In: *Proceedings of the 16th International Conference on World Wide Web*. WWW '07. Banff, Alberta, Canada: Association for Computing Machinery, 2007, pp. 121–130. ISBN: 9781595936547. DOI: `10.1145/1242572.1242590`. URL: `https://doi.org/10.1145/1242572.1242590`.

[31] Kyunghyun Cho. "Noisy parallel approximate decoding for conditional recurrent language model." In: 2016.

[32] Sajal Choudhary, Prerna Srivastava, Lyle H. Ungar, and João Sedoc. "Domain Aware Neural Dialog System." In: vol. abs/1708.00897. 2017.

[33] Elizabeth Clark, Tal August, Sofia Serrano, Nikita Haduong, Suchin Gururangan, and Noah A Smith. "All That's 'Human' Is Not Gold: Evaluating Human Evaluation of Generated Text." In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2021, pp. 7282–7296.

[34] Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A. Smith. "Creative Writing with a Machine in the Loop: Case Studies on Slogans and Stories." In: *23rd International Conference on Intelligent User Interfaces*. IUI '18. Tokyo, Japan: ACM, 2018, pp. 329–340. ISBN: 978-1-4503-4945-1. DOI: `10.1145/3172944.3172983`. URL: `http://doi.acm.org/10.1145/3172944.3172983`.

[35] Andy Coenen, Luke Davis, Daphne Ippolito, Ann Yuan, and Emily Reif. "Wordcraft: a Human-AI Collaborative Editor for Story Writing." In: (2021).

[36] Edith Cohen. *Min-Hash Sketches: A Brief Survey*. 2016. URL: http://www.cohenwang.com/edith/Surveys/minhash.pdf.

[37] Nicole A Cooke. *Fake News and Alternative Facts: Information literacy in a Post-Truth Era*. American Library Association, 2018.

[38] Ning Dai, Jianze Liang, Xipeng Qiu, and Xuanjing Huang. "Style Transformer: Unpaired Text Style Transfer without Disentangled Latent Representation." In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 5997–6007. DOI: 10.18653/v1/P19-1601. URL: https://www.aclweb.org/anthology/P19-1601.

[39] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. "Transformer-xl: Attentive language models beyond a fixed-length context." In: *arXiv preprint arXiv:1901.02860* (2019).

[40] Cristian Danescu-Niculescu-Mizil and Lillian Lee. "Chameleons in Imagined Conversations: A New Approach to Understanding Coordination of Linguistic Style in Dialogs." In: *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 76–87. URL: https://www.aclweb.org/anthology/W11-0609.

[41] Jan Milan Deriu, Don Tuggener, Pius von Däniken, Jon Ander Campos, Álvaro Rodrigo, Thiziri Belkacem, Aitor Soroa, Eneko Agirre, and Mark Cieliebak. "Spot The Bot: A Robust and Efficient Framework for the Evaluation of Conversational Dialogue Systems." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2020, pp. 3971–3984.

[42] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[43] Jesse Dodge, Maarten Sap, Ana Marasovic, William Agnew, Gabriel Ilharco, Dirk Groeneveld, and Matt Gardner. *Documenting the English Colossal Clean Crawled Corpus*. 2021. arXiv: 2104.08758 [cs.CL].

[44] Jesse Dodge, Maarten Sap, Ana Marasovic, William Agnew, Gabriel Ilharco, Dirk Groeneveld, and Matt Gardner. "Documenting the English Colossal Clean Crawled Corpus." In: *arXiv preprint arXiv:2104.08758* (Apr. 2021). arXiv: 2104.08758 [cs.CL].

[45] Chris Donahue, Mina Lee, and Percy Liang. "Enabling Language Models to Fill in the Blanks." In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 2492–2501.

[46] Yao Dou, Maxwell Forbes, Rik Koncel-Kedziorski, Noah A Smith, and Yejin Choi. "Scarecrow: A Framework for Scrutinizing Machine Text." In: *arXiv preprint arXiv:2107.01294* (2021).

[47] Liam Dugan, Daphne Ippolito, Arun Kirubarajan, and Chris Callison-Burch. "RoFT: A Tool for Evaluating Human Detection of Machine-Generated Text." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. 2020, pp. 189–196.

[48] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. "Calibrating noise to sensitivity in private data analysis." In: *Theory of cryptography conference*. Springer. 2006, pp. 265–284.

[49] Angela Fan, Mike Lewis, and Yann Dauphin. "Hierarchical Neural Story Generation." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 889–898. URL: https://www.aclweb.org/anthology/P18-1082.

[50] William Fedus, Barret Zoph, and Noam Shazeer. *Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity*. 2021. arXiv: 2101.03961 [cs.LG].

[51] Vitaly Feldman and Chiyuan Zhang. "What neural networks memorize and why: Discovering the long tail via influence estimation." In: *Advances in Neural Information Processing Systems*. 2020.

[52] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. "Model inversion attacks that exploit confidence information and basic countermeasures." In: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 2015, pp. 1322–1333.

[53] Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. "Style Transfer in Text: Exploration and Evaluation." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2018. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17015.

[54] Rodney A. Gabriel, Tsung-Ting Kuo, Julian McAuley, and Chun-Nan Hsu. "Identifying and characterizing highly similar notes in big clinical note datasets." In: *Journal of Biomedical Informatics* 82 (2018), pp. 63–69. ISSN: 1532-0464. DOI: https://doi.org/10.1016/j.jbi.2018.04.009. URL: https://www.sciencedirect.com/science/article/pii/S153204641830073X.

[55] Karan Ganju, Qi Wang, Wei Yang, Carl A Gunter, and Nikita Borisov. "Property inference attacks on fully connected neural networks using permutation invariant representations." In: *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*. 2018, pp. 619–633.

[56] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. "The Pile: An 800GB Dataset of Diverse Text for Language Modeling." In: *arXiv preprint arXiv:2101.00027* (2020).

[57] Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. "The Pile: An 800GB Dataset of Diverse Text for Language Modeling." In: *arXiv preprint arXiv:2101.00027* (2020).

[58] Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III au2, and Kate Crawford. *Datasheets for Datasets*. 2020. arXiv: 1803.09010 [cs.DB].

[59] Sebastian Gehrmann, Hendrik Strobelt, and Alexander M Rush. "GLTR: Statistical Detection and Visualization of Generated Text." In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. 2019, pp. 111–116.

[60] Kevin Gimpel, Dhruv Batra, Chris Dyer, and Gregory Shakhnarovich. "A systematic exploration of diversity in machine translation." In: *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. 2013, pp. 1100–1111. URL: https://ttic.uchicago.edu/~gregory/papers/emnlp2013diversity.pdf.

[61] David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. "English gigaword." In: *Linguistic Data Consortium, Philadelphia* 4.1 (2003), p. 34.

[62] Jiatao Gu, Kyunghyun Cho, and Victor O.K. Li. "Trainable Greedy Decoding for Neural Machine Translation." In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Copenhagen, Denmark: Association for Computational Linguistics, Sept. 2017, pp. 1968–1978. DOI: 10.18653/v1/D17-1210. URL: https://www.aclweb.org/anthology/D17-1210.

[63] Mandy Guo, Zihang Dai, Denny Vrandecic, and Rami Al-Rfou. "Wiki-40B: Multilingual Language Model Dataset." In: *LREC 2020*. 2020. URL: http://www.lrec-conf.org/proceedings/lrec2020/pdf/2020.lrec-1.296.pdf.

[64] Bikash Gyawali, Lucas Anastasiou, and Petr Knoth. "Deduplication of Scholarly Documents using Locality Sensitive Hashing and Word Embeddings." In: *Proceedings of the 12th Language Resources and Evaluation Conference*. 2020, pp. 901–910.

[65] Tatsunori B. Hashimoto, Hugh Zhang, and Percy Liang. "Unifying Human and Statistical Evaluation for Natural Language Generation." In: *CoRR* abs/1904.02792 (2019). arXiv: 1904.02792. URL: http://arxiv.org/abs/1904.02792.

[66] Jie He, Bo Peng, Yi Liao, Qun Liu, and Deyi Xiong. "TGEA: An Error-Annotated Dataset and Benchmark Tasks for TextGeneration from Pretrained Language Models." In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. 2021, pp. 6012–6025.

[67] Peter Henderson, Koustuv Sinha, Nicolas Angelard-Gontier, Nan Rosemary Ke, Genevieve Fried, Ryan Lowe, and Joelle Pineau. *Ethical Challenges in Data-Driven Dialogue Systems*. 2017. arXiv: 1711.09050 [cs.CL].

[68] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory." In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[69] Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. "The Curious Case of Neural Text Degeneration." In: *CoRR* abs/1904.09751 (2019).

[70] Zhiqiang Hu, Roy Ka-Wei Lee, and Charu C. Aggarwal. "Text Style Transfer: A Review and Experiment Evaluation." In: *CoRR* abs/2010.12742 (2020). arXiv: 2010.12742. URL: https://arxiv.org/abs/2010.12742.

[71] Yichen Huang, Yizhe Zhang, Oussama Elachqar, and Yu Cheng. "INSET: Sentence Infilling with INter-SEntential Transformer." In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 2502–2515.

[72] Daphne Ippolito, Daniel Duckworth, Chris Callison-Burch, and Douglas Eck. "Automatic Detection of Generated Text is Easiest when Humans are Fooled." In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 1808–1822.

[73] Daphne Ippolito, David Grangier, Chris Callison-Burch, and Douglas Eck. "Unsupervised hierarchical story infilling." In: *Proceedings of the First Workshop on Narrative Understanding*. 2019, pp. 37–43.

[74] Daphne Ippolito, Reno Kriz, João Sedoc, Maria Kustikova, and Chris Callison-Burch. "Comparison of Diverse Decoding Methods from Conditional Language Models." In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 2019, pp. 3752–3762.

[75] Paul Jaccard. "The distribution of the flora in the alpine zone." In: *New phytologist* 11.2 (1912), pp. 37–50.

[76] Matthew Jagielski, Jonathan Ullman, and Alina Oprea. "Auditing differentially private machine learning: How private is private sgd?" In: *arXiv preprint arXiv:2006.07709* (2020).

[77] Bargav Jayaraman and David Evans. "Evaluating differentially private machine learning in practice." In: *28th {USENIX} Security Symposium ({USENIX} Security 19)*. 2019, pp. 1895–1912.

[78] Di Jin, Zhijing Jin, Zhiting Hu, Olga Vechtomova, and Rada Mihalcea. "Deep Learning for Text Style Transfer: A Survey." In: *CoRR* abs/2011.00416 (2020). arXiv: 2011.00416. URL: https://arxiv.org/abs/2011.00416.

[79] Zhijing Jin, Di Jin, Jonas Mueller, Nicholas Matthews, and Enrico Santus. "IMaT: Unsupervised Text Attribute Transfer via Iterative Matching and Translation." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3097–3109. DOI: 10.18653/v1/D19-1306. URL: https://www.aclweb.org/anthology/D19-1306.

[80] Anjuli Kannan and Oriol Vinyals. "Adversarial evaluation of dialogue models." In: *arXiv preprint arXiv:1701.08198* (2017).

[81] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. "Scaling laws for neural language models." In: *arXiv preprint arXiv:2001.08361* (2020).

[82] Juha Kärkkäinen and Peter Sanders. "Simple linear work suffix array construction." In: *International colloquium on automata, languages, and programming*. Springer. 2003, pp. 943–955.

[83] Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. "Ctrl: A conditional transformer language model for controllable generation." In: *arXiv preprint arXiv:1909.05858* (2019).

[84] Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. "OpenNMT: Open-Source Toolkit for Neural Machine Translation." In: *Proc. ACL*. 2017. DOI: 10.18653/v1/P17-4012. URL: https://doi.org/10.18653/v1/P17-4012.

[85] Pang Ko and Srinivas Aluru. "Space efficient linear time construction of suffix arrays." In: *Annual Symposium on Combinatorial Pattern Matching*. Springer. 2003, pp. 200–210.

[86] Jonathan Krause, Justin Johnson, Ranjay Krishna, and Li Fei-Fei. "A hierarchical approach for generating descriptive image paragraphs." In: *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*. IEEE. 2017, pp. 3337–3345.

[87] Kalpesh Krishna, John Wieting, and Mohit Iyyer. "Reformulating Unsupervised Style Transfer as Paraphrase Generation." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 737–762. DOI: 10.18653/v1/2020.emnlp-main.55. URL: https://www.aclweb.org/anthology/2020.emnlp-main.55.

[88] Reno Kriz, João Sedoc, Marianna Apidianaki, Carolina Zheng, Gaurav Kumar, Eleni Miltsakaki, and Chris Callison-Burch. "Complexity-Weighted Loss and Diverse Reranking for Sentence Simplification." In: 2019.

[89] Taku Kudo and John Richardson. "SentencePiece: A simple and language independent subword tokenizer and detokenizer for Neural Text Processing." In: *CoRR* abs/1808.06226 (2018). arXiv: 1808.06226. URL: http://arxiv.org/abs/1808.06226.

[90] Ilya Kulikov, Alexander H Miller, Kyunghyun Cho, and Jason Weston. "Importance of a Search Strategy in Neural Dialogue Modelling." In: 2018.

[91] Quoc Le and Tomas Mikolov. "Distributed Representations of Sentences and Documents." In: *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*. ICML'14. Beijing, China, 2014, pp. 1188–1196.

[92] Chris van der Lee, Albert Gatt, Emiel van Miltenburg, Sander Wubben, and Emiel Krahmer. "Best practices for the human evaluation of automatically generated text." In: *Proceedings of the 12th International Conference on Natural Language Generation*. 2019, pp. 355–368.

[93] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. "Deduplicating Training Data Makes Language Models Better." In: 2022.

[94] Brian Lester, Rami Al-Rfou, and Noah Constant. "The Power of Scale for Parameter-Efficient Prompt Tuning." In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 2021, pp. 3045–3059.

[95] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension." In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 2020, pp. 7871–7880.

[96] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. "A Diversity-Promoting Objective Function for Neural Conversation Models." In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 110–119. DOI: 10.18653/v1/N16-1014. URL: https://www.aclweb.org/anthology/N16-1014.

[97] Jiwei Li and Dan Jurafsky. "Mutual information and diverse decoding improve neural machine translation." In: 2016.

[98] Jiwei Li, Will Monroe, and Dan Jurafsky. "A simple, fast diverse decoding algorithm for neural generation." In: 2016.

[99] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. "Adversarial learning for neural dialogue generation." In: *arXiv preprint arXiv:1701.06547* (2017).

[100] Juncen Li, Robin Jia, He He, and Percy Liang. "Delete, Retrieve, Generate: a Simple Approach to Sentiment and Style Transfer." In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 1865–1874. DOI: 10.18653/v1/N18-1169. URL: https://www.aclweb.org/anthology/N18-1169.

[101] Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. "CommonGen: A Constrained Text Generation Challenge for Generative Commonsense Reasoning." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*. 2020, pp. 1823–1840.

[102] Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. "Adversarial ranking for language generation." In: *Advances in Neural Information Processing Systems*. 2017, pp. 3155–3165.

[103] Stephanie Lin, Jacob Hilton, and Owain Evans. *TruthfulQA: Measuring How Models Mimic Human Falsehoods*. 2021. arXiv: 2109.07958 [cs.CL].

[104] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. "Microsoft coco: Common objects in context." In: *European conference on computer vision*. Springer. 2014, pp. 740–755. URL: https://link.springer.com/chapter/10.1007/978-3-319-10602-1_48.

[105] Dayiheng Liu, Jie Fu, Yidan Zhang, Chris Pal, and Jiancheng Lv. "Revision in continuous space: Unsupervised text style transfer without adversarial learning." In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 05. 2020, pp. 8376–8383. URL: https://arxiv.org/abs/1905.12304.

[106] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing." In: *arXiv preprint arXiv:2107.13586* (2021). URL: https://arxiv.org/pdf/2107.13586.pdf.

[107] Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. "Towards an Automatic Turing Test: Learning to Evaluate Dialogue Responses." In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2017, pp. 1116–1126.

[108] Fuli Luo, Peng Li, Jie Zhou, Pengcheng Yang, Baobao Chang, Xu Sun, and Zhifang Sui. "A Dual Reinforcement Learning Framework for Unsupervised Text Style Transfer." In: *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*. Ed. by Sarit Kraus. ijcai.org, 2019, pp. 5116–5122. DOI: 10.24963/ijcai.2019/711. URL: https://doi.org/10.24963/ijcai.2019/711.

[109] Ruotian Luo. *An Image Captioning codebase in PyTorch*. https://github.com/ruotianluo/ImageCaptioning.pytorch. 2017.

[110] Thang Luong, Hieu Pham, and Christopher D. Manning. "Effective Approaches to Attention-based Neural Machine Translation." In: *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal, 2015, pp. 1412–1421. URL: https://aclweb.org/anthology/D15-1166.

[111] Udi Manber and Gene Myers. "Suffix arrays: a new method for on-line string searches." In: *siam Journal on Computing* 22.5 (1993), pp. 935–948.

[112] Javier Marin, Aritro Biswas, Ferda Ofli, Nicholas Hynes, Amaia Salvador, Yusuf Aytar, Ingmar Weber, and Antonio Torralba. "Recipe1M+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images." In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2019).

[113] Stephen Mayhew, Klinton Bicknell, Chris Brust, Bill McDowell, Will Monroe, and Burr Settles. "Simultaneous translation and paraphrase for language education." In: *Proceedings of the Fourth Workshop on Neural Generation and Translation*. 2020, pp. 232–243.

[114] R. Thomas McCoy, Paul Smolensky, Tal Linzen, Jianfeng Gao, and Asli Celikyilmaz. "How much do language models copy from their training data? Evaluating linguistic novelty in text generation using RAVEN." In: *CoRR* abs/2111.09509 (2021). arXiv: 2111.09509. URL: https://arxiv.org/abs/2111.09509.

[115] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient estimation of word representations in vector space." In: *arXiv preprint arXiv:1301.3781* (2013).

[116] Remi Mir, Bjarke Felbo, Nick Obradovich, and Iyad Rahwan. "Evaluating Style Transfer for Text." In: *CoRR* abs/1904.02295 (2019). arXiv: 1904.02295. URL: http://arxiv.org/abs/1904.02295.

[117] Yusuke Mori, Hiroaki Yamane, Yusuke Mukuta, and Tatsuya Harada. "Finding and Generating a Missing Part for Story Completion." In: *Proceedings of the The 4th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*. 2020, pp. 156–166.

[118] Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. "A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories." In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 839–849. DOI: 10.18653/v1/N16-1098. URL: https://www.aclweb.org/anthology/N16-1098.

[119] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. *WebGPT: Browser-assisted question-answering with human feedback*. 2021. arXiv: 2112.09332 [cs.CL].

[120] Ramesh Nallapati, Bowen Zhou, Cícero Nogueira dos Santos, Ãaglar GülÃ§ehre, and Bing Xiang. "Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond." In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL)*. Berlin, Germany, 2016, pp. 280–290.

[121] Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Çağlar Gulçehre, and Bing Xiang. "Abstractive Text Summarization using Sequence-to-sequence RNNs and Beyond." In: *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*. 2016, pp. 280–290.

[122] Milad Nasr, Shuang Song, Abhradeep Thakurta, Nicolas Papernot, and Nicholas Carlini. "Adversary instantiation: Lower bounds for differentially private machine learning." In: *arXiv preprint arXiv:2101.04535* (2021).

[123] Colin Raffel Nikhil Kandpal Eric Wallace. "Deduplicating Training Data Mitigates Privacy Risks in Language Models." In: *arXiv preprint* (2021).

[124] Ge Nong, Sen Zhang, and Wai Hong Chan. "Linear suffix array construction by almost pure induced-sorting." In: *2009 data compression conference*. IEEE. 2009, pp. 193–202.

[125] Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. "Why We Need New Evaluation Metrics for NLG." In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017, pp. 2241–2252.

[126] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. *Carbon Emissions and Large Neural Network Training*. 2021. arXiv: 2104.10350 [cs.LG].

[127] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "GloVe: Global Vectors for Word Representation." In: *Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1532–1543. URL: http://www.aclweb.org/anthology/D14-1162.

[128] Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. "Style Transfer Through Back-Translation." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, 2018, pp. 866–876. DOI: 10.18653/v1/P18-1080. URL: https://www.aclweb.org/anthology/P18-1080.

[129] Raul Puri and Bryan Catanzaro. "Zero-shot text classification with generative language models." In: *arXiv preprint arXiv:1912.10165* (2019). URL: https://arxiv.org/abs/1912.10165.

[130] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. "Improving language understanding by generative pre-training." In: (2018).

[131] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. "Language models are unsupervised multitask learners." In: *OpenAI Blog* 1.8 (2019).

[132] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

[133] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." In: *Journal of Machine Learning Research* 21.140 (2020), pp. 1–67. URL: http://jmlr.org/papers/v21/20-074.html.

[134] Swaroop Ramaswamy, Om Thakkar, Rajiv Mathews, Galen Andrew, H Brendan McMahan, and Françoise Beaufays. "Training production language models without memorizing user data." In: *arXiv preprint arXiv:2009.10031* (2020).

[135] Sudha Rao and Joel Tetreault. "Dear Sir or Madam, May I Introduce the GYAFC Dataset: Corpus, Benchmarks and Metrics for Formality Style Transfer." In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 129–140. DOI: 10.18653/v1/N18-1012. URL: https://www.aclweb.org/anthology/N18-1012.

[136] Emily Reif, Daphne Ippolito, Ann Yuan, Andy Coenen, Chris Callison-Burch, and Jason Wei. "A Recipe for Arbitrary Text Style Transfer with Large Language Models." In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. 2022.

[137] Laria Reynolds and Kyle McDonell. *Prompt Programming for Large Language Models: Beyond the Few-Shot Paradigm*. 2021. arXiv: 2102.07350 [cs.CL].

[138] Parker Riley, Noah Constant, Mandy Guo, Girish Kumar, David C. Uthus, and Zarana Parekh. "TextSETTR: Label-Free Text Style Extraction and Tunable Targeted Restyling." In: *Proceedings of the Annual Meeting of the Association of Computational Linguistics (ACL)* (2021). arXiv: 2010.03802. URL: https://arxiv.org/abs/2010.03802.

[139] A Roberts and C Raffel. "Exploring Transfer Learning with T5: The Text-To-Text Transfer Transformer." In: *Google AI Blog* (2020). URL: https://ai.googleblog.com/2020/02/exploring-transfer-learning-with-t5.html.

[140] Melissa Roemmele. "Inspiration through Observation: Demonstrating the Influence of Automatically Generated Text on Creative Writing." In: (2021), pp. 52–461. URL: https://computationalcreativity.net/iccc21/wp-content/uploads/2021/09/ICCC_2021_paper_32.pdf.

[141] Keisuke Sakaguchi and Benjamin Van Durme. "Efficient Online Scalar Annotation with Bounded Support." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 208–218. DOI: 10.18653/v1/P18-1020. URL: https://www.aclweb.org/anthology/P18-1020.

[142] Evan Sandhaus. "The new york times annotated corpus." In: *Linguistic Data Consortium, Philadelphia* 6.12 (2008), e26752.

[143] Timo Schick and Hinrich Schütze. "It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners." In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, June 2021, pp. 2339–2352. URL: https://www.aclweb.org/anthology/2021.naacl-main.185.

[144] Tal Schuster, Roei Schuster, Darsh J Shah, and Regina Barzilay. "Are We Safe Yet? The Limitations of Distributional Features for Fake News Detection." In: *arXiv preprint arXiv:1908.09805* (2019).

[145] Abigail See, Aneesh Pappu, Rohun Saxena, Akhila Yerukola, and Christopher D Manning. "Do Massively Pretrained Language Models Make Better Storytellers?" In: *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*. 2019, pp. 843–861.

[146] Rico Sennrich, Barry Haddow, and Alexandra Birch. "Neural Machine Translation of Rare Words with Subword Units." In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. DOI: 10.18653/v1/P16-1162. URL: https://www.aclweb.org/anthology/P16-1162.

[147] Noam Shazeer and Mitchell Stern. "Adafactor: Adaptive learning rates with sublinear memory cost." In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4596–4604.

[148] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. "Style Transfer from Non-Parallel Text by Cross-Alignment." In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/2d2c8394e31101a261abf1784302bf75-Paper.pdf.

[149] Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. "Towards controllable biases in language generation." In: *arXiv preprint arXiv:2005.00268* (2020).

[150] Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. "Membership inference attacks against machine learning models." In: *2017 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2017, pp. 3–18.

[151] Irene Solaiman, Miles Brundage, Jack Clark, Amanda Askell, Ariel Herbert-Voss, Jeff Wu, Alec Radford, and Jasmine Wang. "Release Strategies and the Social Impacts of Language Models." In: *arXiv preprint arXiv:1908.09203* (2019).

[152] Jonghyuk Song, Sangho Lee, and Jong Kim. "Crowdtarget: Target-based detection of crowdturfing in online social networks." In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM. 2015, pp. 793–804.

[153] Cory Stephenson, Suchismita Padhy, Abhinav Ganesh, Yue Hui, Hanlin Tang, and SueYeon Chung. "On the geometry of generalization and memorization in deep neural networks." In: *International Conference on Learning Representations*. 2021.

[154] Emma Strubell, Ananya Ganesh, and Andrew McCallum. *Energy and Policy Considerations for Deep Learning in NLP*. 2019. arXiv: 1906.02243 [cs.CL].

[155] Akhilesh Sudhakar, Bhargav Upadhyay, and Arjun Maheswaran. "“Transforming” Delete, Retrieve, Generate Approach for Controlled Text Style Transfer." In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 3269–3279. DOI: 10.18653/v1/D19-1322. URL: https://www.aclweb.org/anthology/D19-1322.

[156] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to Sequence Learning with Neural Networks." In: *Advances in Neural Information Processing Systems 27*. Ed. by Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger. Curran Associates, Inc., 2014, pp. 3104–3112. URL: http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

[157] Yik-Cheung Tam, Jiachen Ding, Cheng Niu, and Jie Zhou. "Cluster-based Beam Search for Pointer-Generator Chatbot Grounded by Knowledge." In: *Dialog System Technology Challenges 7 at AAAI 2019*. 2019.

[158] Wilson L Taylor. "“Cloze procedure”: A new tool for measuring readability." In: *Journalism Bulletin* 30.4 (1953), pp. 415–433.

[159] Om Thakkar, Swaroop Ramaswamy, Rajiv Mathews, and Françoise Beaufays. *Understanding Unintended Memorization in Federated Learning*. 2020. arXiv: 2006.07490 [cs.LG].

[160] Aleena Thomas, David Ifeoluwa Adelani, Ali Davody, Aditya Mogadala, and Dietrich Klakow. "Investigating the impact of pre-trained word embeddings on memorization in neural networks." In: *International Conference on Text, Speech, and Dialogue*. Springer. 2020, pp. 273–281.

[161] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. "LaMDA: Language Models for Dialog Applications." In: *arXiv preprint arXiv:2201.08239* (2022).

[162] Jörg Tiedemann. "News from OPUS-A collection of multilingual parallel corpora with tools and interfaces." In: *Recent advances in natural language processing*. Vol. 5. 2009, pp. 237–248.

[163] Trieu H Trinh and Quoc V Le. "A simple method for commonsense reasoning." In: *arXiv preprint arXiv:1806.02847* (2018).

[164] Alan Turing. "Computing machinery and intelligence-AM Turing." In: *Mind* 59.236 (1950), p. 433.

[165] Chris J Vargo, Lei Guo, and Michelle A Amazeen. "The agenda-setting power of fake news: A big data analysis of the online media landscape from 2014 to 2016." In: *New media & society* 20.5 (2018), pp. 2028–2049.

[166] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention Is All You Need." In: *CoRR* abs/1706.03762 (2017). arXiv: 1706.03762. URL: http://arxiv.org/abs/1706.03762.

[167] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[168] Yannick Versley and Yana Panchenko. "Not just bigger: Towards better-quality Web corpora." In: *Proceedings of the seventh Web as Corpus Workshop (WAC7)*. 2012, pp. 44–52.

[169] Ashwin K Vijayakumar, Michael Cogswell, Ramprasath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. "Diverse Beam Search: Decoding Diverse Solutions from Neural Sequence Models." In: 2016.

[170] Oriol Vinyals and Quoc Le. "A Neural Conversational Model." In: *arXiv preprint arXiv:1506.05869* (2015).

[171] Soroush Vosoughi, Deb Roy, and Sinan Aral. "The spread of true and false news online." In: *Science* 359.6380 (2018), pp. 1146–1151.

[172] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. "Universal adversarial triggers for attacking and analyzing NLP." In: *arXiv preprint arXiv:1908.07125* (2019).

[173] Ben Wang and Aran Komatsuzaki. *GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model*. https://github.com/kingoflolz/mesh-transformer-jax. May 2021.

[174] Gang Wang, Christo Wilson, Xiaohan Zhao, Yibo Zhu, Manish Mohanlal, Haitao Zheng, and Ben Y Zhao. "Serf and turf: crowdturfing for fun and profit." In: *Proceedings of the 21st international conference on World Wide Web*. ACM. 2012, pp. 679–688.

[175] Ryan Webster, Julien Rabin, Loïc Simon, and Frédéric Jurie. "Detecting Overfitting of Deep Generative Networks via Latent Recovery." In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 11265–11274. DOI: 10.1109/CVPR.2019.01153.

[176] Peter Weiner. "Linear pattern matching algorithms." In: *14th Annual Symposium on Switching and Automata Theory (swat 1973)*. IEEE. 1973, pp. 1–11.

[177] Orion Weller, Nicholas Lourie, Matt Gardner, and Matthew E. Peters. "Learning from Task Descriptions." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, Nov. 2020, pp. 1361–1375. DOI: 10.18653/v1/2020.emnlp-main.105. URL: https://www.aclweb.org/anthology/2020.emnlp-main.105.

[178] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. Rush. "Transformers: State-of-the-Art Natural Language Processing." In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. URL: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[179] Yonghui Wu. "Smart compose: Using neural networks to help write emails." In: *Google AI Blog* (2018).

[180] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. "Google's neural machine translation system: Bridging the gap between human and machine translation." In: *arXiv preprint arXiv:1609.08144* (2016).

[181] Jingjing Xu, Xu Sun, Qi Zeng, Xiaodong Zhang, Xuancheng Ren, Houfeng Wang, and Wenjie Li. "Unpaired Sentiment-to-Sentiment Translation: A Cycled Reinforcement Learning Approach." In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 979–988. DOI: 10.18653/v1/P18-1090. URL: https://www.aclweb.org/anthology/P18-1090.

[182] Peng Xu, Yanshuai Cao, and Jackie Chi Kit Cheung. "On Variational Learning of Controllable Representations for Text without Supervision." In: *Proceedings of the International Conference on Machine Learning (ICML)* abs/1905.11975 (2020). arXiv: 1905.11975. URL: http://arxiv.org/abs/1905.11975.

[183] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. "mT5: A massively multilingual pre-trained text-to-text transformer." In: *arXiv preprint arXiv:2010.11934* (2020).

[184] Mikio Yamamoto and Kenneth W Church. "Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus." In: *Computational Linguistics* 27.1 (2001), pp. 1–30.

[185] Samuel Yeom, Irene Giacomelli, Matt Fredrikson, and Somesh Jha. "Privacy risk in machine learning: Analyzing the connection to overfitting." In: *2018 IEEE 31st Computer Security Foundations Symposium (CSF)*. IEEE. 2018, pp. 268–282.

[186] Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito. "Wordcraft: Story Writing With Large Language Models." In: *27th International Conference on Intelligent User Interfaces*. 2022, pp. 841–852.

[187] Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. "Defending Against Neural Fake News." In: *arXiv preprint arXiv:1905.12616* (2019).

[188] Wei Zeng, Xiaozhe Ren, Teng Su, Hui Wang, Yi Liao, Zhiwei Wang, Xin Jiang, ZhenZhang Yang, Kaisheng Wang, Xiaoda Zhang, Chen Li, Ziyan Gong, Yifan Yao, Xinjing Huang, Jun Wang, Jianfeng Yu, Qi Guo, Yue Yu, Yan Zhang, Jin Wang, Hengtao Tao, Dasen Yan, Zexuan Yi, Fang Peng, Fangqing Jiang, Han Zhang, Lingfeng Deng, Yehong Zhang, Zhe Lin, Chao Zhang, Shaojie Zhang, Mingyue Guo, Shanzhi Gu, Gaojun Fan, Yaowei Wang, Xuefeng Jin, Qun Liu, and Yonghong Tian. "PanGu-α: Large-scale Autoregressive Pretrained Chinese Language Models with Auto-parallel Computation." In: *arXiv preprint arXiv:2104.12369* (2021).

[189] Chiyuan Zhang, Daphne Ippolito, Katherine Lee, Matthew Jagielski, Florian Tramèr, and Nicholas Carlini. "Counterfactual Memorization in Neural Language Models." In: *arXiv preprint arXiv:2112.12938* (2021).

[190] Hugh Zhang, Daniel Duckworth, Daphne Ippolito, and Arvind Neelakantan. "Trading Off Diversity and Quality in Natural Language Generation." In: *Proceedings of the Workshop on Human Evaluation of NLP Systems (HumEval)*. 2021, pp. 25–33.

[191] Xiang Zhang, Junbo Zhao, and Yann LeCun. "Character-Level Convolutional Networks for Text Classification." In: *Proceedings of the Conference on Neural Information Processing Systems* (Sept. 2015). arXiv: 1509.01626 [cs].

[192] Yizhe Zhang, Michel Galley, Jianfeng Gao, Zhe Gan, Xiujun Li, Chris Brockett, and Bill Dolan. "Generating informative and diverse conversational responses via adversarial information maximization." In: *Advances in Neural Information Processing Systems*. 2018, pp. 1815–1825. URL: https://papers.nips.cc/paper/7452-generating-informative-and-diverse-conversational-responses-via-adversarial-information-maximization.pdf.

[193] Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. *Calibrate Before Use: Improving Few-Shot Performance of Language Models*. 2021. arXiv: 2102.09690 [cs.CL].

[194] Wanrong Zhu, Zhiting Hu, and Eric Xing. "Text infilling." In: *arXiv preprint arXiv:1901.00158* (2019).

[195] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books." In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 19–27.

[196] Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. "A Monolingual Tree-based Translation Model for Sentence Simplification." In: *Proceedings of the 23rd International Conference on Computational Linguistics (COLING 2010)*. Beijing, China: Coling 2010 Organizing Committee, Aug. 2010, pp. 1353–1361. URL: https://www.aclweb.org/anthology/C10-1152.

[197] Albert Ziegler. *GitHub Copilot: Parrot or Crow?* https://docs.github.com/en/github/copilot/research-recitation. 2021.

[198] Jakub Łącki, Vahab Mirrokni, and Michał Włodarczyk. *Connected Components at Scale via Local Contractions*. 2018. arXiv: 1807.10727 [cs.DC].