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

© COPYRIGHT

2021

Daphne Ippolito

# ACKNOWLEDGEMENT

Understanding the Limitations of Using

Large Language Models for Text Generation A B S T R A C T

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

Improvements to neural language modelling over the last half-decade have led to systems which are capable of generating incredibly fluent text. Natural languages generation systems have been successfu deployed in many practical applications, from machine translation and keyboard predicting to text summarization and specification.

This dissertation addresses two main questions.

- What problems arise when attempting to use language models for generation in real-world settings?

- How can neural language models be used to enable creativity

The first problem I focus on is the ability to generate text that humans perceive as high quality but has the same diversity of language present in genuine human writing.

The second problem is on what happens when generated text is **{TODO: }**. Are humans able to notice when they are reading language model-generated text?

The third problem is the privacy implications of training on large, barely curated text datasets extracted from the internet.

## 1.1 GOALS OF THIS WORK

something something something

# 2 | BACKGROUND ON NEURAL LANGUAGE MODELS

## 2.1 WHAT IS A LANGUAGE MODEL?

## 2.2 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 produce probability distribution over the next word in sequence given the previous words. Neural language models have the advantage that they assign non-zero probabilities to sequences never seen in the training data, and thus they can handle very long context sequences.

Rather than computing probabilities for sequences of discrete words, the first step of a neural language model is to embed each word into a continuous vector space. An embedding matrix (whose rows are the embeddings for each word in the vocabulary) is used to map a sequence of words to a sequence of fixed-sized real-numbered vectors. Let $\mathbf{y}_1, \ldots, \mathbf{y}_n$ be the sequence of embeddings for some text. Typical neural language models emit $\hat{\mathbf{y}}_t$, a predicted embedding for the $t$th position in the sequence given the previous embeddings in the sequence. This can be written as

$$\hat{\mathbf{y}}_t = f(\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) \tag{2.1}$$

where $f$ is the neural network and $\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}$ are the embeddings of the previous tokens in the sequence. To train the weights in $f$ (and the weights of the embedding matrix), we measure how how close the emitted embedding $\hat{\mathbf{y}}_t$ is to the embedding of the true next word.

Some language models condition both on the previous words of the target sequence as well as an additional sequence of text. This paradigm is known as an encoder-decoder or sequence-to-sequence, and the function above is modified to

$$\hat{\mathbf{y}}_t = f(\mathbf{y}_1, \ldots, \mathbf{y}_{t-1}; \mathbf{x}_1, \ldots, \mathbf{x}_n) \tag{2.2}$$

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 previous words of the English sequence.

Finally, to produce a probability distribution for what the next word should be, the predicted embedding $\hat{\mathbf{x}}_t$ is multiplied by the embedding matrix $\mathbf{E}$ (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 $X_t$ be a random variable representing the vocabulary item predicted for the $t$th position. We then have:

$$P(Y_t = i | \mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) = \frac{\exp(\mathbf{E}\hat{\mathbf{y}}_t[i])}{\sum_j \exp(\mathbf{E}\hat{\mathbf{y}}_t[j])} \tag{2.3}$$

where $i$ and $j$ are indexes into the vocabulary.

Neural language models are almost always trained with a log-likelihood loss that encourages the model to put more probability mass on the true next token than the alternatives:

$$\mathcal{L} = -\sum_{t=1}^{T} \log P(Y_t = i^* | \mathbf{y}_1, \ldots, \mathbf{y}_{t-1}) \tag{2.4}$$

where $i^*$ is the label of the of the true next token.

Most state-of-the-art work uses a variant of the Transformer architecture [83] as the neural network $f$. Except for my work described in Section **{TODO: decide on section}**, which uses an LSTM-based model, all my research uses the Transformer model architecture.

For simplicity, this section refers to the input to a language model as a sequence of words, but it in practice most state-of-the-art systems divide text into a sequence of sub-word tokens, such as those produced by a Byte Pair Encoding (BPE) [70]. Typical BPE vocabularies range from 32k to 50k words. Only my work in Section **{TODO: decide on section}** uses an older word-level vocabulary.

## 2.3 GENERATING TEXT WITH A LANGUAGE MODEL

Language models in themselves are not generative. AS noted in the previous section, they only predict a probability distribution for what the next token in the sequence *could* be. To perform generation, an algorithm for decoding sequences from these predicted probability distributions is needed. At each step of decoding, the 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 decoding strategies for selecting a token from the prediction probability distribution. The simplest strategy is to take the $\arg\max$ of the distribution. 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. Lastly, search methods such as beam search can be used to traverse many possible likely sequences and pick the best one overall. The diversity-quality tradeoffs involved in search-based and random sampling decoding strategies are described in Sections {TODO: } and {TODO: } respectively.

## 2.4    NEURAL LANGUAGE MODELS AND CREATIVITY

Story generation and creative writing assistance have been {TODO: ()} of natural language processing since the early days of the field. Since I started my PhD in 2015, language generation systems have gone from largely rule-based systems able to generate highly structured text in narrow domains {TODO: citation} to multi-billion parameter language models capable of generating text which human raters are often unable to distinguish from text written by human writers {TODO: citation}.

This progress has led many to try and understand the potential impact of natural language generation systems on {TODO: }. There are two over-arching questions whch bear consideration. Are neural language models useful for generating novel and engaging stories and other creative writing, either when used standalone or as a tool in the hands of human writers? What are the ethical considerations around using a neural network, especially one trained on the writings of real authors, to try and replicate human creativity?

# 3

# MEASURING GENERATED TEXT QUALITIY AND THE DIVERSITY-QUALITY TRADEOFF

## 3.1 MOTIVATION

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, we present investigations of how the choice of decoding startegy 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.

## 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 [37]. Others consider diversity a measure of how different two or more sentences are from each other [85, 32]. 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 [59]. 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 [77, 57], text summarization [61], and dialog systems [86].

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 [77]. 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 [50].

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 [19]. 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. [44] 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 [49, 50, 32]. Furthermore, in open-ended dialog, the use of reranking to personalize a model's responses for each user is a promising research direction [18].

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 [49, 85, 46, 78]. 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 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.

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 [50]. 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 [17], 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

---

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.

---

**Algorithm 1** Beam Search Inference

---

 1: **procedure** BEAM SEARCH
 2:     $B \leftarrow \{SOS\}$
 3:     $k \leftarrow$ BeamWidth
 4:     $out \leftarrow k$-best output list
 5:     **while** $|out| < k$ **do**
 6:         $front \leftarrow$ remove all nodes from $B$
 7:         **for** $w \in front$ **do**
 8:             $succ \leftarrow w$'s $k$-best successors
 9:             **for** $s \in succ$ **do**
10:                 **if** $s == EOS$ **then**
11:                     $out \leftarrow out \cup \{s\}$
12:                 **else**
13:                     $B \leftarrow B \cup \{s\}$
14:                 **end if**
15:             **end for**
16:         **end for**
17:         Sort $B$
18:         **if** $|B| > k$ **then**
19:             Prune $B$ to $k$-best successors
20:         **end if**
21:     **end while**
        **return** out
22: **end procedure**

---

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 $\sigma_0$ 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 [34]. 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 [50] and Li et al. [51], 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. [85] proposes adding an additional diversity-promoting term, $\theta$, 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 $i$th candidate at timestep $t$ becomes:

$$\text{score}(c_{\leq t}^{(i)}) = \sum_{j=1}^{t} \left( \log P(c_j^{(i)}|c_{<j}^{(i)}, \mathbf{x}) \right)$$

$$+ \lambda \theta(c_{\leq t}^{(i)}, \mathcal{H}_{t-1})$$

where $\lambda$ is a tunable hyperparameter. Vijayakumar et al. [85] try a variety of definitions for $\theta$, 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 $\theta$.

**ITERATIVE BEAM SEARCH**    In an attempt to improve the size of the search space explored without sacrificing runtime, Kulikov et al. [46] 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 $i$th 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 $\epsilon$ Hamming distance from a previously explored state are also excluded. During the experimentation of Kulikov et al. [46], 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. [78] 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. [45] then clustered similar sentences together to further increase the variety of simplifications from which to choose. Document embeddings generated via Paragraph Vector [47] 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) [23].[3] Second, after clustering, Kriz et al. [45] 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

---

2  We follow Tam et al. [78] and used averaged GloVe word embeddings [65].
3  BERT sentence-level embeddings were obtained using https://github.com/hanxiao/bert-as-service.

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.

### 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 [42]. We match the model architecture and training data of Baheti et al. [7]. 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 [79] to predict the next utterance given the previous one.

Evaluation is performed on 100 prompts from the Cornell Movie Dialog Corpus [22]. These prompts are a subset of the 1000 prompts used in Baheti et al. [7], 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 **??** show 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 3.1:** 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. [5]. We take advantage of Luo [56]'s open-source implementation and released model parameters trained on MSCOCO [54]. 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[4]. 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. [85] 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

| | | SPICE | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Method** | | **Mean** | **@1** | **@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 3.2:** 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.

the candidate captions, which is useful to know for applications that combine diverse candidate output sequences [44].

*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 [49]. 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. [97] instead propose an entropy metric, **Ent-k**, defined as:

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

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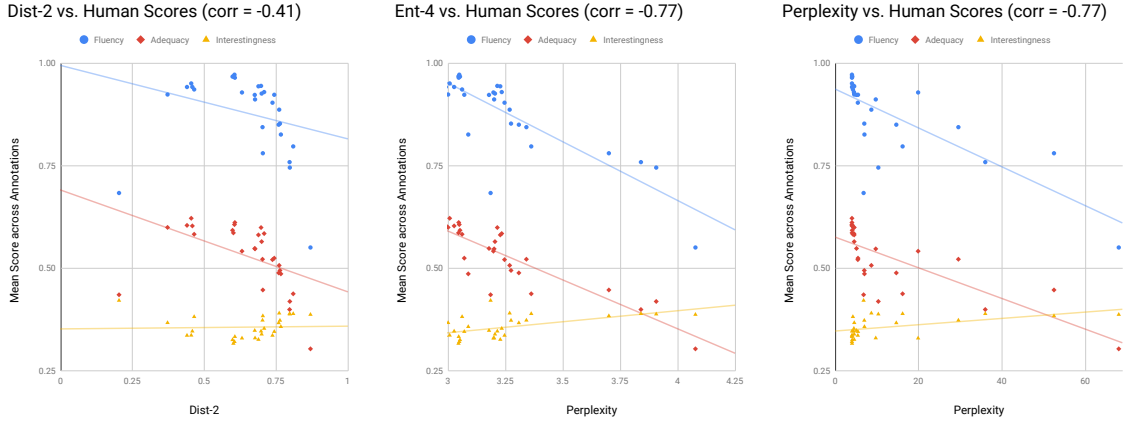
**Figure** 3.2: Each decoding strategy is plotted, showing that human-perceived quality is negatively correlated with diversity. The Pearson Correlation coefficients between each statistic and the average of fluency, coherence, and interestingness are shown in parentheses.

### *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 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 do-

main, 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.[5]

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.

---

5  In the appendix, we show 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.

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

## 3.4    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 [95, 66, 1]. 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 [**cooke2018fake**]. It propogates quickly [87], sets political agendas [82], influences elections [3], and undermines user trust [**song2015crowdtarget**, 89]. Recently, Adelani et al. [1] 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 [81]. It continues to be used is chatbot evaluation

[55]. 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 [48].

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 [66]. We draw a large number of excerpts generated by each strategy and train a family of BERT-based [23] 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 [95]. 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 [31]. **bakhtin2019real** 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. [68] 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 [53, 52] and in automatic evaluation of generative model quality [63, 40, 55].

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 [66]. 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. [38] 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 [26]. 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 $p$ [38].

We thus consider three different decoding strategy settings:

- Sample from the untruncated distribution
- Top-$k$, choosing $k$=40 [66].
- Nucleus sampling (aka top-$p$), choosing $p$=0.96 [95].

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.[6] 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 [92] (excerpts might be quite short if the model produces an end-of-text token early on). See Appendix 1 for final dataset sizes.

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

---

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

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 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 [92]. 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 [23]. 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 [70], 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 [31], 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. [74] 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. (Full instructions are in the Appendix.) 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 (see Appendix Table 4) as a group. Afterward, they were asked to complete 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 the Appendix.

(a)

(b)
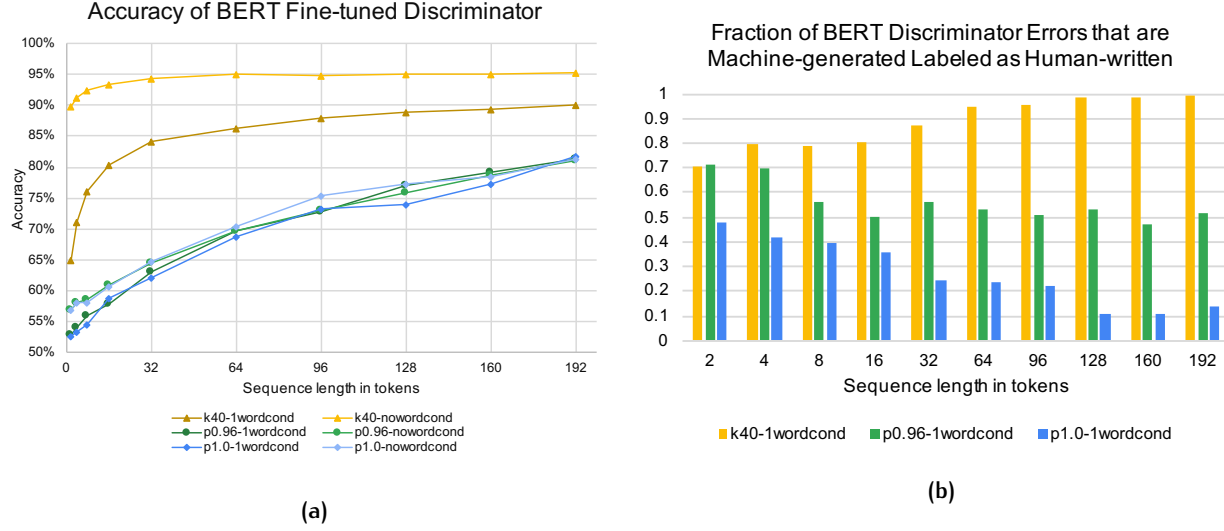
**Figure** 3.3: In **(a)**, accuracy increases as the length of the sequences used to train the discriminator is increased. In **(b)**, we see that the BERT fine-tuned discriminator predicts about the same number of false-positives as false-negatives when trained with samples generated using top-$p$ 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.

### 3.4.3 Results

SIMPLE BASELINES    Table 3.4 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.

Logistic regression on bag-of-words is the best of the baselines, beating out the histogram-based methods. While Gehrmann et al. [31] report an AUC of 0.87 on classifying text as real or generated using logistic regression on the four buckets of the GLTR system,

(a)

(b)
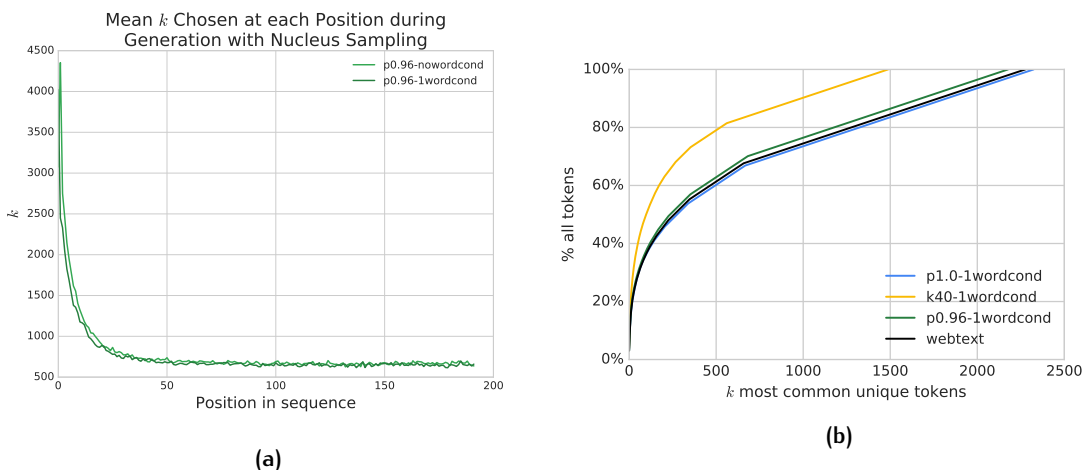
Figure 3.4: In **(a)**, the average (over sequences in the test set) $k$ chosen at each step during generating with nucleus sampling is plotted. Adding a single word of priming strongly impacts the $k$s chosen for the first few positions, but this difference quickly dissipates. In **(b)**, we consider 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.

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 3.3a, 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 3.4b, 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. [69] 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.[7] 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 3.3a). 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 3.4a where at the very first step of unprimed generation, nucleus sampling selects from 3075 possible vocabulary words, and

---

[7] when decoding from the GPT-2 small model with 117M parameters.
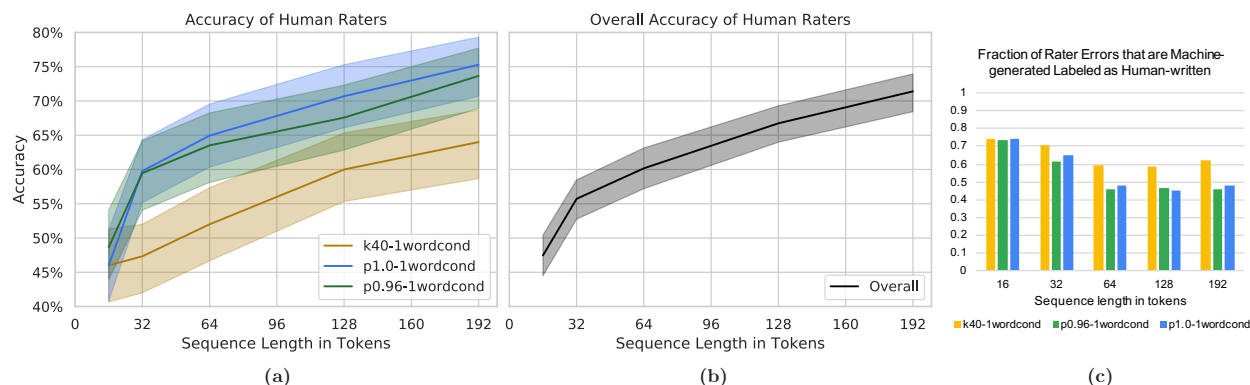
Figure 3.5: **(a)** and **(b)** show human rater accuracy of correctly identifying an excerpt as human-written or machine-written, shown with 80% confidence internals, in **(a)**, broken up by decoding strategy and in **(b)**, overall. Accuracy increases as raters observe more tokens. **(c)** shows that 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.

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 3.5, 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. Unsurprisingly 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 3.6, 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 3.3b), 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 3.5b. 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 3.5a. 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. [95] 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 3.7 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 Conclusion

In this work, we study the behavior of automated discriminators and their ability to identify machine-generated and human-written texts. We train these discriminators on

balanced binary classification datasets where all machine-generated excerpts are drawn from the same generative model but with different decoding strategies. We find that, in general, discriminators transfer poorly between decoding strategies, but that training on a mix of data from methods can help. We also show the rate at which discriminator accuracy increases as excerpts are lengthened.

We further study the ability of expert human raters to perform the same task. We find that rater accuracy varies wildly, but has a median of 74%, which is less than the accuracy of our best-performing discriminator. Most interestingly, we find that human raters and discriminators make decisions based on different qualities, with humans more easily noticing semantic errors and discriminators picking up on statistical artifacts. In our experiments, these artifacts are most prominent with top-$k$ sampling. However, any strategy that over-samples high-likelihood words is susceptible. As the $p$ in nucleus sampling is set increasingly lower to achieve more fluent text (some systems are already using $p$ as low as 0.5 [60]), the distributional deviations that plague top-$k$ text will surface in nucleus sampling as well.

Holtzman et al. [38] explain how a unique attribute of human language is that it dips in and out of low probability zones. This variance in likelihood is what makes human-written text interesting and exciting to read. Today's generation systems have not yet solved the problem of mimicking the human cadence without introducing poor word choices that are easy for humans to detect. Generation systems often optimize for fooling humans without acknowledging the trade-off that exists between human perception of quality and ease of automatic detection. We therefore suggest three prongs for future research:

1. Identifying ways to improve the language models and decoding strategies we use in order to generate text that is both exciting (ie. unlikely) and semantically plausible.

2. Building better world understanding into automatic discriminators so that they are more capable of detecting the types of errors that humans notice.

3. Developing tools and educational materials to improve humans' ability to detect machine-generated text. These may include automatic detectors with components that explain their predictions.

Finally, we would like to note that all of our experiments were performed with English language models, and it remains an open question how the trade-off between ease of human detection and ease of automatic detection might differ for languages that are very different from English.

## 3.5 DETECTING THE BOUNDARY BETWEEN HUMAN−WRITTEN AND MACHINE− GENERATED TEXT

| **Prompt:** Look, nobody knows we did it. | |
|---|---|
| **RS 0.5** | **Standard BS** |
| I don't know what you're talking about. | We've got to get out of here. |
| What's the matter with you? | What do you mean? |
| I don't know what it is. | I don't think it's a good idea. |
| I don't think so. | I don't know what to say. |
| He's got to get out of here. | I don't know what's going on. |
| **RS 1.0** | **Standard BS with PDC** |
| I can't find it. | I don't know! |
| They're our ships. | I don't think so. |
| It's all right anyone is the right to interfere. | What do you mean? |
| We didn't have a plan I engineered a policy. | Why didn't you tell me? |
| Same time you pick us up at six and get we. | That's why we're here. |
| **RS 1.0,top10** | **RS 1.0,top10 with PDC** |
| I don't know what else to do. | What do you mean? |
| It doesn't have to be that way! | I don't think so. |
| We're in the air! | That's why I'm here. |
| I've seen a guy in his place in a it. | It's all right we. |
| And I'm not we any more. | We've been through this before. |
| **NPAD0.3 BS** | **Cluster5 BS** |
| I don't think it's a good idea. | I don't know why. |
| I don't know what to say. | What do you mean? |
| I don't know what's going on. | I don't think so. |
| I don't know what to do. | How do you know that? |
| I don't know what's going on here. | 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 3.3: Responses to an example prompt for selected methods. More examples can be seen in the appendix.

| Method | BERT | | BagOfWords | | HistGLTR | | Hist50Buckets | | TotalProb | Human |
|---|---|---|---|---|---|---|---|---|---|---|
| | 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 3.4: 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 (1worc-cond). 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.

| | | 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 3.5: 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 3.6: 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 3.7:** 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.

# 4 | DETECTION OF MACHINE–GENERATED TEXT

## 4.1 MOTIVATION

## 4.2 BACKGROUND

## 4.3 METHOD

## 4.4 RESULTS

How much can a Transformer model actually memorize? One experiment we could do is to take the original train set and split it into two, A and B. Deduplicate the n-grams between them (so if an n-gram is in A, it won't be in B). Then train/finetune a Transformer on the binary classification task: given an n-gram predict yes if it is in A and no if it is not in A.

# 5 | MEMORIZATION

## 5.1 MOTIVATION

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 [15, 93, 33, 13]. 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 [10, 88, 72]. Quantitatively and qualitatively understanding these datasets is therefore a research challenge in its own right [24].

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 (§5.4). Using a suffix array [58], 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 [12], 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 (§5.5).

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 §5.6.4)—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 $10\times$.

2. Train-test overlap is common in non-deduplicated datasets. For example, we find *a 61-word sequence*[8] in C4 [67] 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.

---

8 "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!"

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 [10, 76, 64].

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 [66, 67], by training on higher quality data the models can reach higher accuracy faster.

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 §5.6. We then examine the impact of deduplication on test perplexity (§5.6.4) and on the frequency of emitting memorized content (§5.6.4). 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 (§5.6.4).

## 5.2 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 [83] 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. [66] 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 [13] with the addition of book datasets, GROVER [95] on a restricted subset filtered to news domains called RealNews, and T5 [67] on a cleaned version of common crawl called C4. Other models are trained on more curated Internet sources—for example Guo et al. [35] 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. [96] for instance introduced PANGU-$\alpha$, 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 [15], a smaller dataset commonly used for evaluation.

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. [66, §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. [25] 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 [80, Appendix B] removed documents from their CommonCrawl-based train set that overlapped substantially with the commonsense reasoning used for evaluation. And GPT-3 [13, §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. [14]. 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. 6, 90, 27, 75]

DUPLICATE TEXT IN TRAINING DATA.    The Book Corpus [98], which was used to train popular models such as BERT, has a substantial amount of exact-duplicate documents according to Bandy and Vincent [8]. Allamanis [2] shows that duplicate examples in code datasets cause worsened performance on code understanding tasks.

## 5.3    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 [35]. 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 [15]. 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. [66, 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 [67]. 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. [24] for a detailed breakdown of the source text in C4.

**REALNEWS**　　is a subset of the Common Crawl consisting of articles from news domains [95]. 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 [11] and then excluding any document which resulted in a hash collision. Like C4, examples with duplicate URLs were excluded.

## 5.4 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.

### 5.4.1  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 [58] is a representation of a suffix tree [91] that can be constructed in linear time in $\|\mathcal{S}\|$ [41] 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$\times$ more memory efficient [58], 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 [94] and document clustering [16].

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 (§5.6.4).

### 5.4.2   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.

### 5.4.3   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.

### 5.4.4   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. [62] and Ko and Aluru [43]. 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[9] 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

---

9 https://github.com/BurntSushi/suffix

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 $S = S_1 \mathbin{||} S_2$ because of the boundary conditions. Instead, we don't build the data $S = S_1 \mathbin{||} S_2$ but rather let $S'_1 = S_1 \mathbin{||} S_2[uptoK]$ 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 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.

### 5.4.5 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 $8\times$ 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.

## 5.5    METHOD FOR APPROXIMATE MATCHING WITH MINHASH

### 5.5.1    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,

| 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! |

**Table 5.1:** 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.

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 5.1).

MinHash [12] is an approximate matching algorithm widely used in large-scale deduplication tasks [84, 28, 36], including to deduplicate the training set for a large Chinese-language LM [96]. 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* [39]:

$$\text{Jaccard}(d_i, d_j) = |d_i \cap d_j| / |d_i \cup d_j| \tag{5.1}$$

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 [20].

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

$$\Pr(d_i, d_j | \operatorname{Jaccard}(d_i, d_j) = s_{i,j}) = 1 - (1 - s_{i,j}^b)^r \tag{5.2}$$

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:

$$\operatorname{EditSim}(x_i, x_j) = 1 - \frac{\operatorname{EditDistance}(x_i, x_j)}{\max(|x_i|, |x_j|)} \tag{5.3}$$

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. [99] to identify connected components.

### 5.5.2 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

$$\Pr(d_i, d_j | \mathrm{Jaccard}(d_i, d_j) = s_{i,j}) = 1 - (1 - s_{i,j}^b)^r \tag{5.4}$$

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.
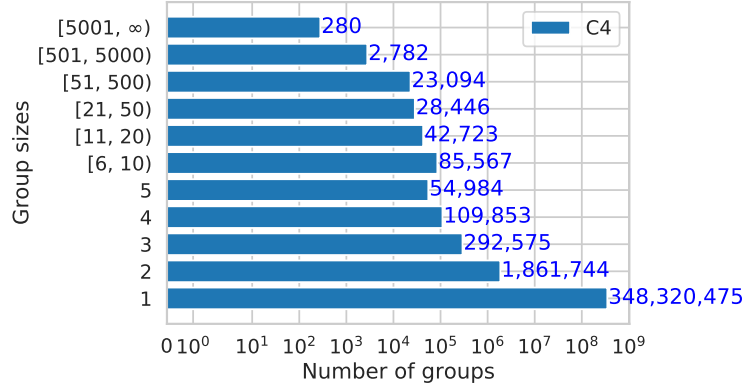
**Figure 5.1:** The distribution of near-duplicate cluster sizes from running NEARDUP on C4.

### 5.5.3 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

$$O(N + bk^2T^2N) = O(N) \tag{5.5}$$

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.

## 5.6 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.

### 5.6.1 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 5.2). Near-duplicate text is much less common in Wiki-40B, forming only 0.39% of the train set.[10] 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 5.1), 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 $8\times$ 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.

---

10 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.

| | % 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 5.2: 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 % |

Table 5.3: The fraction of tokens (note Table 5.2 reports the fraction of *examples*) identified by EXACTSUBSTR as part of an exact duplicate 50-token substring.

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

Table 5.4: 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.

### 5.6.2  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 5.1. Table 5.4 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.

### 5.6.3  Train / Test Set Leakage

Both deduplication methods identify overlap between the train set and the validation set (Table 5.2). 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 5.6.4.

### 5.6.4   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. [67], 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 [71]. 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 5.2). 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 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.

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

Table 5.5: 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.

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 [14]. For our generation experiments, we use top-$k$ random sampling with $k = 50$ and experiment with prompted and unprompted generation.

GENERATION WITH NO PROMPT.      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 5.5). 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 5.6.

GENERATION WITH PROMPTING.      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 5.4 shows the proportion of generations which meet this requirement, while Figure 5.3 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 5.8 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 [21], which was trained on LM1B, and GROVER [95], 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 [**gpt-neo**] are part of a 50 token matches of its training data, the Pile [29].

## 5.7  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 [9, 30], less work has been done to quantitatively analyze properties of real language modelling datasets, like Dodge et al. [24] 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 [73].[11] 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[12], 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

---

11  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.

12  https://play.aidungeon.io/

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.

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.
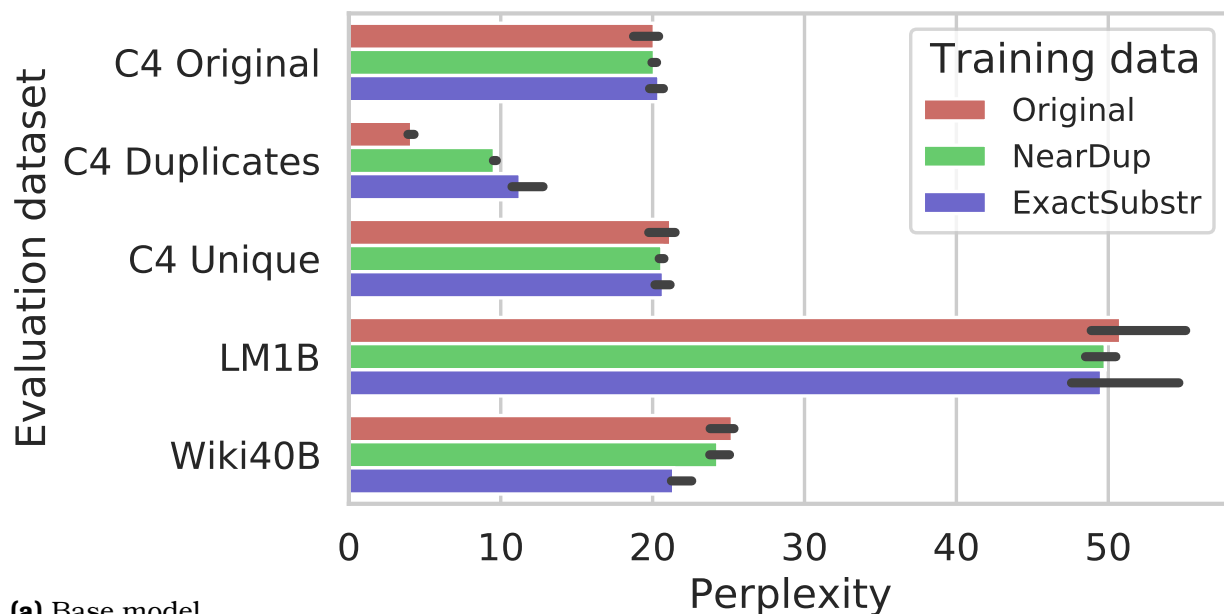
**(a)** Base model
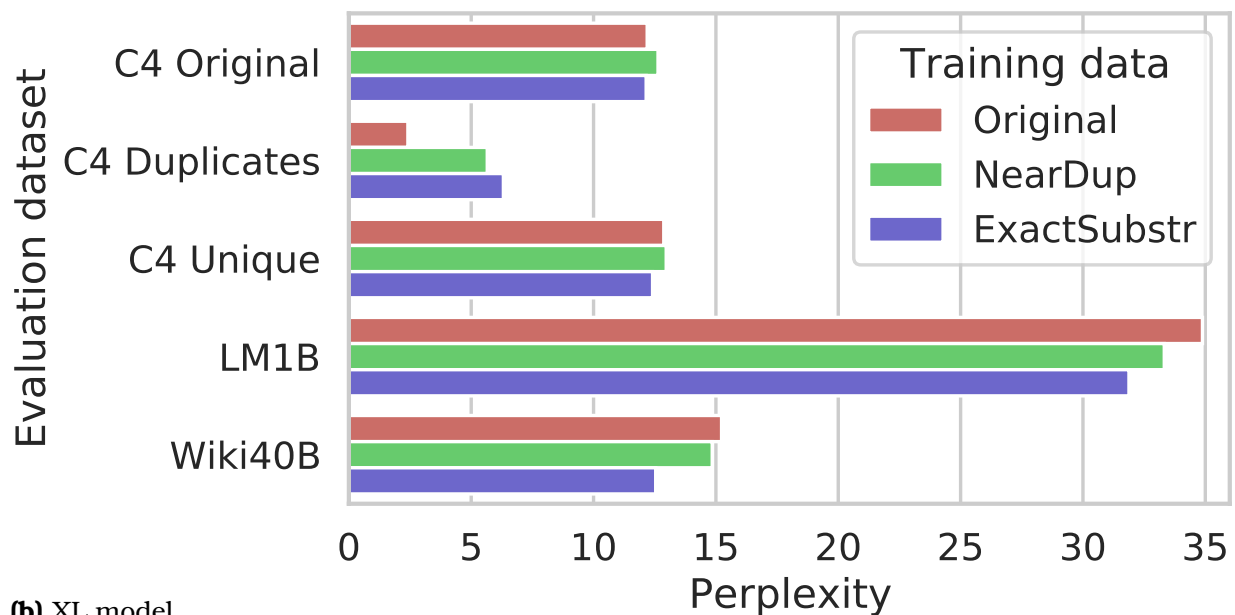


**(b)** XL model

**Figure 5.2:** 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 5.6: 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.

| 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<br>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, BIRTHPLACE: 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<br>f their records, too. Around 1900, the Roman Catholic churches reported | 2 |
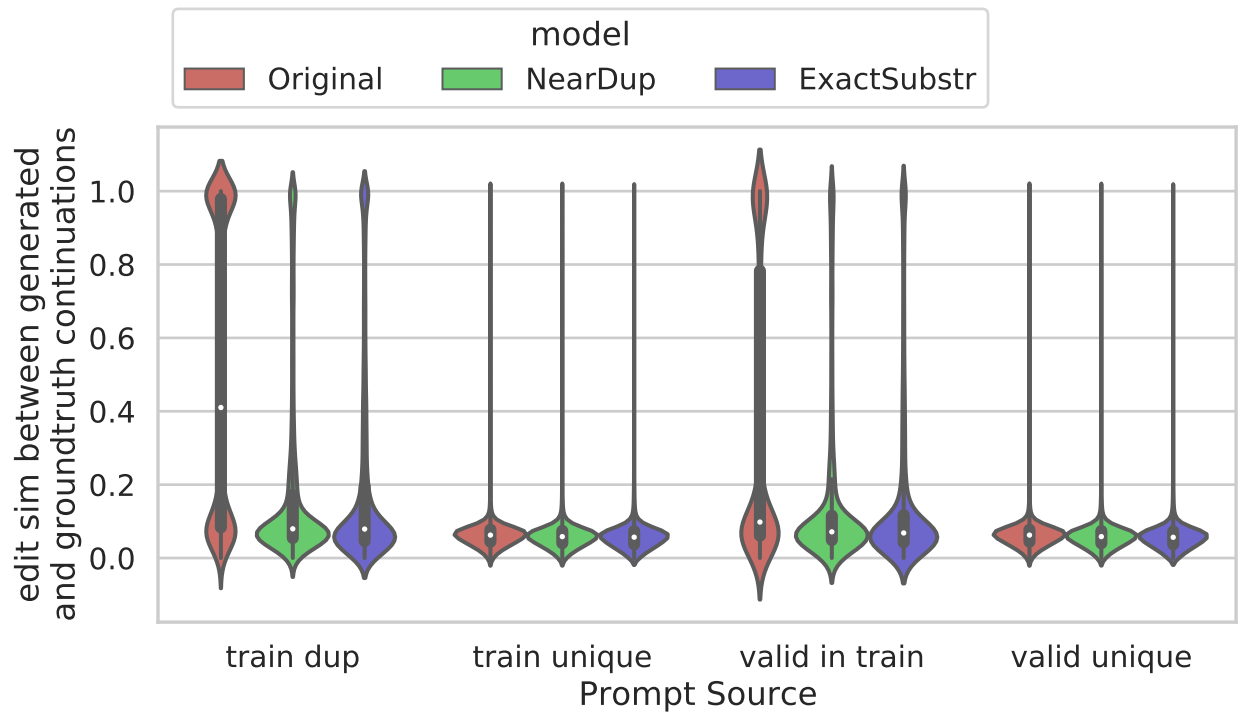
**Figure 5.3:** Memorized continuations distribution
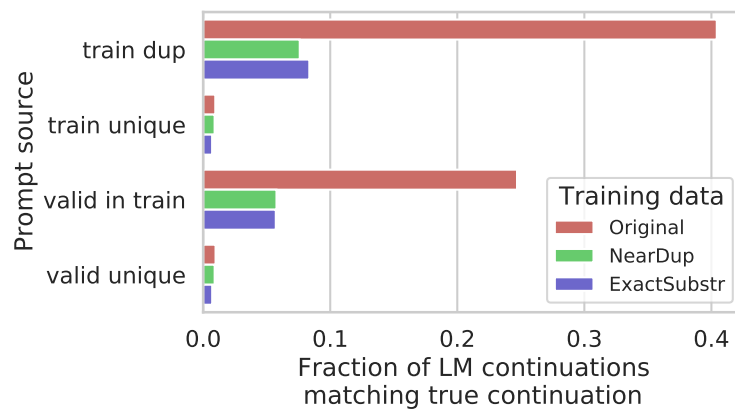


**Figure 5.4:** The proportion of generations which have edit similarity above 0.8 with the groundtruth continuation when using the LM to generate continuations for 32-token prompts identified by NEARDUP as either duplicated or unique.

| 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 |

Table 5.8: 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.

# 6 | FILLLING IN THE BLANK

## 6.1 MOTIVATION

## 6.2 BACKGROUND

## 6.3 METHOD

## 6.4 RESULTS

# 7 | CREATIVITY

## 7.1 MOTIVATION

## 7.2 BACKGROUND

## 7.3 METHOD

## 7.4 RESULTS

# 8 | CONCLUSIONS

I conclude things.

# LIST OF TABLES

# LIST OF ILLUSTRATIONS

# BIBLIOGRAPHY

[1]  David Ifeoluwa Adelani, Haotian Mai, Fuming Fang, Huy H Nguyen, Junichi Ya-magishi, 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.

[2]  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.

[3]  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.

[4]  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.

[5]  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.

[6]  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.

[7]  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.

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

[9]  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.

[10] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?" In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. FAccT 2021. Virtual Event, Canada: Association for Computing Machinery, 2021, pp. 610–623. ISBN: 9781450383097. DOI: 10.1145/3442188.3445922. URL: https://doi.org/10.1145/3442188.3445922.

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

[12] 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.

[13] 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.

[14] 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].

[15] 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).

[16] 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.

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

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

[19] 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.

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

[21]  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).

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

[23]  Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-
      training of deep bidirectional transformers for language understanding." In: 2018.

[24]  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].

[25]  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].

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

[27]  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.

[28]  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.

[29]  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).

[30]  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].

[31]  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.

[32]  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.

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

[34]  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.

[35]  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.

[36]  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.

[37]  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.

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

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

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

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

[42]  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.

[43] 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.

[44] 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.

[45] 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.

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

[47] 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.

[48] 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.

[49] 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.

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

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

[52] 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).

[53] 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.

[54] 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.

[55] 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.

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

[57] 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.

[58] 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.

[59] 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.

[60] Lesly Miculicich, Marc Marone, and Hany Hassan. "Selecting, Planning, and Rewriting: A Modular Approach for Data-to-Document Generation and Translation." In: *EMNLP-IJCNLP 2019* (2019), p. 289.

[61] 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.

[62] 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.

[63] 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.

[64] 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].

[65] 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.

[66] 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).

[67] 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.

[68] 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).

[69] 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.

[70] 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.

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

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

[73] 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.

[74] 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).

[75] 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.

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

[77] 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.

[78]   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.

[79]   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.

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

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

[82]   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.

[83]   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.

[84]   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.

[85]   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.

[86]   Oriol Vinyals and Quoc V. Le. "A Neural Conversational Model." In: vol. abs/1506.05869. 2015. URL: https://arxiv.org/pdf/1506.05869.pdf.

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

[88]   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).

[89]   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.

[90]   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.

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

[92] 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).

[93] 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).

[94] 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.

[95] 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).

[96] Wei Zeng, Xiaozhe Ren, Teng Su, Hui Wang, Yi Liao, Zhiwei Wang, Xin Jiang, Zhen-Zhang 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-$\alpha$: Large-scale Autoregressive Pretrained Chinese Language Models with Auto-parallel Computation." In: *arXiv preprint arXiv:2104.12369* (2021).

[97] 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.

[98] 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.

[99] Jakub Łącki, Vahab Mirrokni, and Michał Włodarczyk. *Connected Components at Scale via Local Contractions*. 2018. arXiv: 1807.10727 [cs.DC].