Deduplicating Training Data Makes Language Models Better

Katherine Lee*† **Daphne Ippolito***†‡ **Andrew Nystrom**† **Chiyuan Zhang**†

Douglas Eck† **Chris Callison-Burch**‡ **Nicholas Carlini**†

https://github.com/google-research/deduplicate-text-datasets

Abstract

We find that existing language modeling datasets contain many near-duplicate examples and long repetitive substrings. As a result, over 1% of the unprompted output of language models trained on these datasets is copied verbatim from the training data. We develop two tools that allow us to deduplicate training datasets—for example removing from C4 a single 61 word English sentence that is repeated over 60,000 times. Deduplication allows us to train models that emit memorized text ten times less frequently and require fewer training steps to achieve the same or better accuracy. We can also reduce train-test overlap, which affects over 4% of the validation set of standard datasets, thus allowing for more accurate evaluation. Code for deduplication is released at `https://github.com/google-research/deduplicate-text-datasets`.

말 그대로의

C4에는 61종류의
한 단어로 구성된
문장들이 6만개나
있음

Dedup하면
더 적은 비용으로
학습이 가능하며
모델도 bias 있는
암기를 하지 않게
되어 더 똑똑해짐

1 Introduction

A key factor behind the recent progress in natural language processing is the development of large-scale text corpora used to train increasingly large language models. These datasets have grown from single gigabytes to as much as a terabyte over the past few years (Chelba et al., 2013; Xue et al., 2020; Graff et al., 2003; Brown et al., 2020). 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 (Bender et al., 2021; Wallace et al., 2019; Sheng et al., 2020). Quantitatively and qualitatively understanding these datasets is therefore a research challenge in its own right (Dodge et al., 2021a).

데이터는 모델에게
내재된 Bias를
만들게 됨

이는 Perplexity나
Valid Loss등으로
잡기 어려움

구석구석 스며드는

We show that one particular source of bias, duplicated training examples, is pervasive: all four common NLP datasets we studied contained duplicates. Additionally, all four corresponding validation sets contained text duplicated in the training set. 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.

We propose two scalable techniques to detect and remove duplicated training data. *Exact* substring matching identifies verbatim strings that are repeated. This allows us to identify cases where only part of a training example is duplicated (§4.1). *Approximate* full document matching uses hash-based techniques (Broder, 1997) to identify pairs of documents with high n-gram overlap (§4.2).

2가지 Dedup 기법 씀

1) Exact substring matching
말그대로 반복되는
문자열을 제거함

2) Approximate full document matching
hash 기반의 기법으로
높은 n-gram으로
중복되는 document
쌍을 식별함

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 §6.2)—even though the 1.5 billion parameter model is much smaller than the 350GB dataset it was trained on. By deduplicating the training dataset we reduce the rate of emitting memorized training data by a factor of 10×.

2. Train-test overlap is common in non-deduplicated datasets. For example, we find a *61-word sequence*[1] in C4 (Raffel et al., 2020) 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).

Dedup을 통해 크게
4가지 이점을 얻을 수
있음

1) 모델은 학습에서
암기된 텍스트를
Generation함
Dedup을 통해 이러한
암기된 텍스트의
Generation이
1 / 10 수준으로 줄어듦

2) C4의 경우 Test,
Training Set에서
하나의 단어로만
이루어진 시퀀스가
있음
이는 모델의 정확도가
과대평가되게 함

[1] "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!"

* Equal contribution. † Google Research, Brain Team. ‡ University of Pennsylvania. Correspond to katherinelee@google.com and daphnei@seas.upenn.edu.

This train-test set overlap not only causes researchers to over-estimate model accuracy, but also biases model selection towards models and hyperparameters that intentionally overfit their training datasets.

3) 데이터셋의 크기가 최대 19%까지 줄어듦 학습 시간에 긍정적임

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 (Bender et al., 2021; Strubell et al., 2019; Patterson et al., 2021).

4) Perplexity가 훼손되지 않음 즉 고품질데이터로 학습하면 적은량의 데이터로 동일한 퍼포먼스를 낼 수 있음

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 (Radford et al., 2019; Raffel et al., 2020), by training on higher quality data the models can reach higher accuracy faster.

Dedup은 장점만 있고 페널티가 없음

To summarize, data duplication offers significant advantages and no observed disadvantages. In the remainder of this paper we present our text deduplication framework in §4, and study the extent of duplicate content in common NLP datasets (e.g., C4, Wiki-40B, and LM1B) in §5. We then examine the impact of deduplication on test perplexity (§6.1) and on the frequency of emitting memorized content (§6.2). 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 (§6.3).

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 (Vaswani et al., 2017) 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. (2019) 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

GPT2 계열은 Reddit 기반으로 학습되었지만 GPT2의 데이터셋이 공개되진 않음

GPT3외 다른 모델들은 CC를 많이 씀

is CommonCrawl, an index of public webpages. Among the models trained on CommonCrawl include GPT-3 (Brown et al., 2020) with the addition of book datasets, GROVER (Zellers et al., 2019) on a restricted subset filtered to news domains called RealNews, and T5 (Raffel et al., 2020) on a cleaned version of common crawl called C4. Other models are trained on more curated Internet sources—for example Guo et al. (2020) 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. (2021) for instance introduced PANGU-α, a family of models with up to 200B parameters that were trained on a non-public corpus of cleaned and filtered Chinese-language documents from CommonCrawl and other sources. Since many of these datasets are not public, we deduplicate three that are: Wiki-40B, C4, and RealNews–as well as the One Billion Word Language Model Benchmark (Chelba et al., 2013), a smaller dataset commonly used for evaluation.

Wiki-40B, C4, RealNews로 Dedup 해봄

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. (2019, §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. (2021b) 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 (2018, Appendix B) removed documents from their CommonCrawl-based train set that overlapped substantially with the commonsense reasoning used for evaluation. And GPT-3 (Brown et al., 2020, §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.

Private한 데이터를 암기하기도 함 이를 막는데도 효과적

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. (2020). 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. Arpit et al., 2017; Webster et al., 2019; Feldman and Zhang, 2020; Stephenson et al., 2021; Teterwak et al., 2021).

중복된 데이터를 학습하면 모델 퍼포먼스가 떨어짐

Duplicate text in training data. The Book Corpus (Zhu et al., 2015), which was used to train popular models such as BERT, has a substantial amount of exact-duplicate documents according to Bandy and Vincent (2021). Allamanis (2019) shows that duplicate examples in code datasets cause worsened performance on code understanding tasks.

3 Language Modeling Datasets

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 (Guo et al., 2020). 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 (Chelba et al., 2013). 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. (2019, 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 (Raffel et al., 2020). 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. (2021a) for a detailed breakdown of the source text in C4.

RealNews is a subset of the Common Crawl consisting of articles from news domains (Zellers et al., 2019). 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 (Bloom, 1970) and then excluding any document which resulted in a hash collision. Like C4, examples with duplicate URLs were excluded.

4 Methods for Identifying Duplicates

The simplest technique to find duplicate examples would be to perform exact string matching between all example pairs, but as we will show, this is insufficient. We introduce two complementary methods for performing deduplication. First, using a suffix array (Manber and Myers, 1993), we remove duplicate substrings from the dataset if they occur verbatim in more than one example. Second, we use MinHash (Broder, 1997), 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.

모든 example의
문자가 exact match
하면 제거하는게
직관적인 Dedup

그러나 이로는 부족함
따라서 추가적인
방법 2가지를 사용

1) Suffix Array
SubString이 1개 이상
중복되면 데이터셋을
제거

2) MinHash
모든 example pair
에서 n-gram 유사도
를 계산

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*: $x_i = \left[x_i^1, x_i^2, \cdots, x_i^{s_i}\right]$.

4.1 Exact Substring Duplication

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. Based on statistical analyses (§B), we select $k = 50$ tokens as the minimum matching substring length.

언어적 특성상
두 Document가
완벽하게 동일할
가능성은 낮음

따라서 Document
내의 일부 문자가
많이 동일 하면
제거하는게 좋음
(EXACTSUBSTR)

본 논문에서는
Document 내에
50토큰의 substr이
일치하면 제거함

A breakdown of the computation needed for this approach can be found in Appendix B.

https://ko.wikipedia.org/wiki/접미사_배열

4.1.1 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 (Manber and Myers, 1993) is a representation of a suffix tree (Weiner, 1973) that can be constructed in linear time in $\|\mathcal{S}\|$ (Kärkkäinen and Sanders, 2003) and enables efficient computation of many substring queries; in particular, they allow us to identify duplicated training examples in linear time. Suffix arrays have the advantage over suffix trees in that they are 10–100× more memory efficient (Manber and Myers, 1993), 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 (Yamamoto and Church, 2001) and document clustering (Chim and Deng, 2007).

exatc-matching은 제곱으로 계산량이 폭증함

따라서 suffix array로 선형적인 시간 복잡도를 구현하여 사용함

Suffix Array는 Suffix tree보다 10 ~ 100배 메모리 효율성이 더 높으나 일부 쿼리에서는 덜 효율적일 수 있음

점근적으로

Suffix Array는 TF-IDF나 Document Clustering 등에서 효율적으로 사용됨

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

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

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 bytes of the BPE tokenization of the text (§6).

4.1.2 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 $\mathcal{A}$.

중복되는 문자열은 정렬된 Suffix Array에서 인접하게 됨

SA[i]와 SA[i+1] ... 에서 중복되는 length를 threshold로 설정하고 인접한 원소들을 중복되는 값으로 기록하면됨

~의 문제

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. This algorithm is embarrassingly parallel, and so we can efficiently process the dataset. Based on experimentation (Appendix B), we choose a threshold length of 50 BPE tokens for all experiments.

본 논문에서는 BPE 기준 50 토큰으로 threshold를 설정함

용이하게

MinHash 설명
https://haandol.github.io/2019/05/25/minhash-algorithm-explained.html

LSH 설명
https://haandol.github.io/2019/05/28/lsh-minhash-explained.html

4.2 Approximate Matching with MinHash

exact 말고 near도 수행

We also perform *approximate* deduplication based on matching entire examples. This method, which we call NEARDUP, is a good complement to the *exact* substring matching, especially for web crawl text, as it handles the very common case of documents being identical except for interspersed templated fields (such as the last row of Table 1).

~사이에 배치하다

MinHash (Broder, 1997) is an approximate matching algorithm widely used in large-scale deduplication tasks (Versley and Panchenko, 2012; Gabriel et al., 2018; Gyawali et al., 2020), including to deduplicate the training set for a large Chinese-language LM (Zeng et al., 2021). 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* (Jaccard, 1912):

n-gram을 사용하여 MinHash 알고리즘 사용

두 document xi, xj를 n-gram 집합인 di, dj로 표현한 뒤 해시함수를 사용하여 자카드 인덱스에 근사함

각각의

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

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 (Cohen, 2016).

di, dj의 자카드 인덱스가 충분히 크다면 두 문서는 유사함

MinHash는 n-gram을 해시함수로 정렬한 뒤 가장 작은 해시된 n-gram k개만 유지하여 signature를 생성함

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

본 논문에서는 5-gram과 signuatre 크기는 9000개로 함

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

where $b = 20$ and $r = 450$ are user-settable parameters to control the strength of the filter. See Appendix A for more details.

b, r은 필터의 강도를 조절하는 사용자 설정 가능한 파라미터임

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:

계산량을 늘려서 더 복잡한 필터를 만들고 정교하게 필터링할 수 있음

이 Edit 유사도가 0.8이상이면 유사하다고 판정

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

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

유사한 문서들의 클러스터를 생성하기 위해 두 문서가 일치하는 경우 두 문서간의 엣지를 연결하는 그래프를 구성함

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 1: Qualitative examples of near-duplicates identified by NEARDUP from each dataset. The similarity 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.

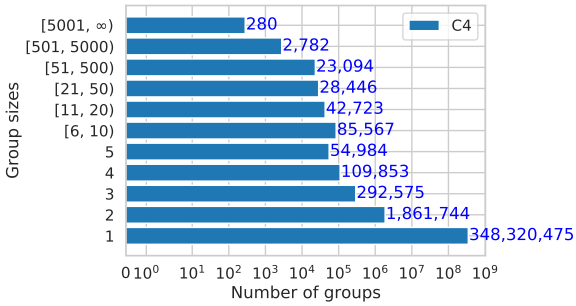


Figure 1: The distribution of near-duplicate cluster sizes from running NEARDUP on C4.

use the method introduced in Łącki et al. (2018) to identify connected components. A breakdown of the computation needed is given in Appendix A.

5 Deduplication 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.

5.1 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 2). Near-duplicate text is much less common in Wiki-40B, forming only 0.39% of the train set.[2] 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 1), including one cluster of size 250,933.

[2]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 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 3: The fraction of tokens (note Table 2 reports the fraction of *examples*) identified by EXACTSUBSTR as part of an exact duplicate 50-token substring.

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

예외) exact의 기준 토큰 50개 미만인 문서가 많아서 LM1B는 exact에서 Dedup이 잘 안되므

5.2 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 1 (more examples in the Appendix).

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.

5.3 Train / Test Set Leakage

Both deduplication methods identify overlap between the train set and the validation set (Table 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 6.3.

6 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. (2020), 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

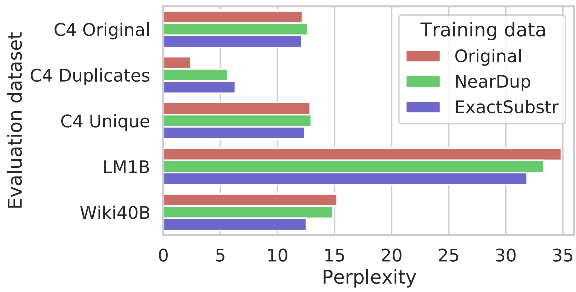


Figure 2: Impact of deduplicating the training set on validation perplexity. We plot the results from T5 XL (see Appendix for base-sized model). For C4, we evaluate on *C4 Original*, the original validation set; *C4 Unique*, a subset of the validation set identified by NEARDUP as having zero matches across C4; and *C4 Duplicates*, a subset of the validation set identified by NEARDUP as having a match in the C4 train set.

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.) Further training details can be found in Appendix C.

6.1 Model Perplexity

exact가 near보다 perplexity가 더 높았음
(물론 threshold에 따라 달라짐)
dedup하면 perplexity가 유사하거나 더 낮아짐

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 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 (for LM1B eval with base models, see Appendix Figure 7). EXACTSUBSTR deduplication decreases perplexity of the XL model by almost 3 points perplexity on Wiki-40B which is

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

Table 4: 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.

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.

6.2 Generated Text

데이터 중복은 모델이 bias를 갖게 하고 다양성이 낮은 generation을 하게 함

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 (Carlini et al., 2020). For our generation experiments, we use top-k random sampling with $k = 50$ and experiment with prompted and unprompted generation.

프롬프트 없이 top-k 50으로 10만개를 실험해봄

1%이상이 암기된 subs sequence를 만들어냄

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 (examples provided in the Appendix). For each generated token, we say the token is memorized if it is part of a 50-token substring that is exactly contained in the training data. On XL-ORIGINAL, over 1% of the generated tokens belong to memorized sub-sequences (see Table 4). 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 9 in the Appendix.

Training에 포함된 데이터의 프롬프트를 제공하면 더 암기함

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 3). 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.

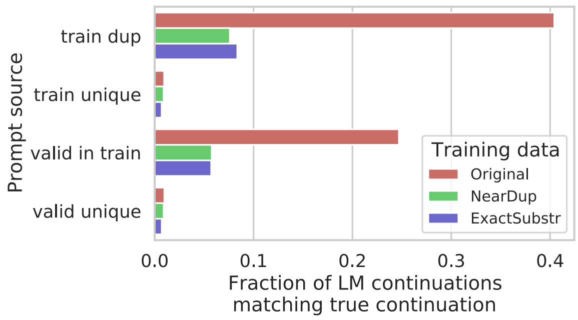


Figure 3: The proportion of generations which have edit similarity above 0.8 with the groundtruth continuation when using the LM to generate continuations for 32-token prompts identified by NEARDUP as either duplicated or unique.

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: 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.

6.3 Impact on Existing Models

Train-test leakage does not just impact models trained on C4. Table 5 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 (Dai et al., 2019), which was trained on LM1B, and GROVER (Zellers et al., 2019), 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 [3] 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 (Black et al., 2021) are part of a 50 token matches of its training data, the Pile (Gao et al., 2020).

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 (Bender and Friedman, 2018; Gebru et al., 2020), less work has been done to quantitatively analyze properties of real language modelling datasets, like Dodge et al. (2021a) 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 (Shokri et al., 2017).[4] 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[5], should also not output memorized content like adverts for real products.

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

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 (Brown et al., 2022).

Dedup의 단점?
Closed Book QA,
Retrieval 등의
작업은 오히려
암기를 요구함

중요한 정보는
오히려 반복되는데
이러한 정보를
학습 못할 수 있음

PII 제거등은
효과적이지 못함

Ultimately, whether memorization is a desired property of a language model, or else risky and unwanted, depends both 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 they have 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.

8 Conclusion

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, dedu-

[3] gs://grover-models/generation_examples/generator=mega~dataset=p0.90.jsonl

[4] 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.

[5] https://play.aidungeon.io/

plication 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.

Ethics

The developers of large language models typically attempt to create training data that reflects natural human communication, but current methods to collect and curate such datasets are fallible. There are multiple reasons some text ends up over-represented. For example, bot replies, auto-generated templates, and licenses are repeated for structural (e.g., legal, economical) reasons (as was also observed by Dodge et al. (2021a)). Additionally, common techniques for acquiring and "cleaning" data can result in an over-representation of particular subsets of world users, often those who are English-speaking and publishing in established forums. This effectively under-represents non-English speakers as well as groups whose communication mostly occurs outside of the public web. In this paper, we focus on the problem of over-representation of some types of text (structural duplicates) but do not address the problem of under-representation of others.

Additionally, while we discuss when memorized content might be desired and when it might not be desired, our analysis does not disambiguate these two cases. Work to disambiguate helpful from harmful memorization is tremendously complex and would require a different set of research methodologies than are presented in this work.

Acknowledgements

We are grateful to the many researchers whose technical help, feedback, and discussions shaped this project: Jacob Austin, Samy Bengio, Olivier Bousquet, James Bradbury, Fernando Diaz, Mark Diaz, Noah Fiedel, Jonathan Frankle, David Grangier, Stefanie Karp, David Mimno, Gaurav Mishra, Michael Mozer, Sharan Narang, Alex Passos, Adam Roberts, Hanie Sedghi, Jascha Sohldickstein, David So, Florian Tramer, and Yun William Yu. We are also grateful to the Google Brain women who have given us continuous support.

Chris Callison-Burch and Daphne Ippolito's research is supported in part by the DARPA KAIROS Program (contract FA8750-19-2-1004), the DARPA LwLL Program (contract FA8750-19-2-0201), and the IARPA BETTER Program (contract 2019-19051600004). The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of DARPA, IARPA, or the U.S. Government.

Contributions

Each of the authors on this paper significantly contributed to the final results.

- Katherine trained the models used in the paper, built and ran the eval and text generation pipelines, contributed significantly to writing, analysis, and project organization and management.
- Daphne ran the approximate matching data deduplication pipelines, extracted prompts and evaluation datasets, ran eval pipelines, and contributed significantly to planning, writing, and analysis.
- Andrew wrote the code to perform deduplication with approximate matching, helped evaluate energy expenditure, and helped with analysis.
- Chiyuan helped generate plots and contributed to project scoping, writing, and data analysis.
- Chris offered mentorship and guidance throughout the project and contributed to writing.
- Doug offered mentorship and guidance throughout the project and contributed to writing.
- Nicholas wrote the suffix array implementation, ran all EXACTSUBSTR deduplication experiments, contributed significantly to planning, writing, and analysis, as well as scoping the project.

References

Miltiadis Allamanis. 2019. 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*, pages 143–153.

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

Jack Bandy and Nicholas Vincent. 2021. Addressing "documentation debt" in machine learning research: A retrospective datasheet for bookcorpus.

Emily M. Bender and Batya Friedman. 2018. Data statements for natural language processing: Toward mitigating system bias and enabling better science. *Transactions of the Association for Computational Linguistics*, 6:587–604.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? 🦜. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Sid Black, Leo Gao, Phil Wang, Connor Leahy, and Stella Biderman. 2021. GPT-Neo: Large scale autoregressive language modeling with mesh-tensorflow.

Burton H Bloom. 1970. Space/time trade-offs in hash coding with allowable errors. *Communications of the ACM*, 13(7):422–426.

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

Hannah Brown, Katherine Lee, Fatemehsadat Mireshghallah, Reza Shokri, and Florian Tramèr. 2022. What does it mean for a language model to preserve privacy? *arXiv preprint*.

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

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. 2020. Extracting training data from large language models.

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

Hung Chim and Xiaotie Deng. 2007. A new suffix tree similarity measure for document clustering. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 121–130, New York, NY, USA. Association for Computing Machinery.

Edith Cohen. 2016. Min-hash sketches: A brief survey.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Jesse Dodge, Maarten Sap, Ana Marasovic, William Agnew, Gabriel Ilharco, Dirk Groeneveld, and Matt Gardner. 2021a. Documenting the english colossal clean crawled corpus.

Jesse Dodge, Maarten Sap, Ana Marasovic, William Agnew, Gabriel Ilharco, Dirk Groeneveld, and Matt Gardner. 2021b. Documenting the english colossal clean crawled corpus. *arXiv preprint arXiv:2104.08758*.

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

Rodney A. Gabriel, Tsung-Ting Kuo, Julian McAuley, and Chun-Nan Hsu. 2018. Identifying and characterizing highly similar notes in big clinical note datasets. *Journal of Biomedical Informatics*, 82:63–69.

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. 2020. The Pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Timnit Gebru, Jamie Morgenstern, Briana Vecchione, Jennifer Wortman Vaughan, Hanna Wallach, Hal Daumé III au2, and Kate Crawford. 2020. Datasheets for datasets.

David Graff, Junbo Kong, Ke Chen, and Kazuaki Maeda. 2003. English gigaword. *Linguistic Data Consortium, Philadelphia*, 4(1):34.

Mandy Guo, Zihang Dai, Denny Vrandecic, and Rami Al-Rfou. 2020. Wiki-40b: Multilingual language model dataset. In *LREC 2020*.

Bikash Gyawali, Lucas Anastasiou, and Petr Knoth. 2020. Deduplication of scholarly documents using locality sensitive hashing and word embeddings. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 901–910.

Paul Jaccard. 1912. The distribution of the flora in the alpine zone. *New phytologist*, 11(2):37–50.

Juha Kärkkäinen and Peter Sanders. 2003. Simple linear work suffix array construction. In *International colloquium on automata, languages, and programming*, pages 943–955. Springer.

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

Udi Manber and Gene Myers. 1993. Suffix arrays: a new method for on-line string searches. *siam Journal on Computing*, 22(5):935–948.

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

David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

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

Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2020. Towards controllable biases in language generation. *arXiv preprint arXiv:2005.00268*.

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

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

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in nlp.

Piotr Teterwak, Chiyuan Zhang, Dilip Krishnan, and Michael C Mozer. 2021. Understanding invariance via feedforward inversion of discriminatively trained classifiers. In *International Conference on Machine Learning*, pages 10225–10235. PMLR.

Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

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

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*.

Ryan Webster, Julien Rabin, Loïc Simon, and Frédéric Jurie. 2019. Detecting overfitting of deep generative networks via latent recovery. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11265–11274.

Peter Weiner. 1973. Linear pattern matching algorithms. In *14th Annual Symposium on Switching and Automata Theory (swat 1973)*, pages 1–11. IEEE.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2020. mt5: A massively multilingual pre-trained text-to-text transformer. *arXiv preprint arXiv:2010.11934*.

Mikio Yamamoto and Kenneth W Church. 2001. Using suffix arrays to compute term frequency and document frequency for all substrings in a corpus. *Computational Linguistics*, 27(1):1–30.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. *arXiv preprint arXiv:1905.12616*.

Wei Zeng, Xiaozhe Ren, Teng Su, Hui Wang, Yi Liao, Zhiwei Wang, Xin Jiang, ZhenZhang Yang, Kaisheng Wang, Xiaoda Zhang, Chen Li, Ziyan Gong, Yifan Yao, Xinjing Huang, Jun Wang, Jianfeng Yu, Qi Guo, Yue Yu, Yan Zhang, Jin Wang, Hengtao Tao, Dasen Yan, Zexuan Yi, Fang Peng, Fangqing Jiang, Han Zhang, Lingfeng Deng, Yehong Zhang, Zhe Lin, Chao Zhang, Shaojie Zhang, Mingyue Guo, Shanzhi Gu, Gaojun Fan, Yaowei Wang, Xuefeng Jin, Qun Liu, and Yonghong Tian. 2021. Pangu-α: Large-scale autoregressive pretrained chinese language models with auto-parallel computation. *arXiv preprint arXiv:2104.12369*.

Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the IEEE international conference on computer vision*, pages 19–27.

Jakub Łącki, Vahab Mirrokni, and Michał Włodarczyk. 2018. Connected components at scale via local contractions.

MinHash 개념

1. shingling(혹은 n-gram)

- 문서를 일정한 크기(n)의 토큰 묶음(shingle = n-gram)의 집합으로 변환
- 예: n=2(2-gram)일 때, “I love cats” -> Shingles = { “I love”, “love cats” }

2. 집합 표현(Set of Shingles)

- 문서를 여러 shingles의 집합으로 표현함

3. MinHash Signature 생성

- 여러 종류의 해시함수(랜덤하게 정의함)들을 각각의 Shingle에 적용함
- 각 문서의 Shingle 집합 중, 각 해시 함수 h1, h2, h3...을 적용한 결과 해시값 중 가장 작은 해시값(Minimum Hash Value)를 뽑아 Signature에 기록
- 보통 해시 함수를 k개 정의하면, 문서마다 길이가 k인 MinHash Signature가 생성됨
  * 3개의 해시함수 적용(k=3) -> [Min(h1(x1), h1(x2) ...), Min(h2(x1), h2(x2) ...), Min(h3(x1), h3(x2) ...) ...]

4. Jacaard 유사도 근사

- 두 문서의 MinHash Signature를 비교했을 때, 동일한 위치에서 해시값이 일치하는 비율이 문서 간의 Jacaard 유사도 근사값임
- Jacaard 유사도 근사값이 Threshold 이상이면 Near-Duplicate로 판단함

MinHash 예시

1. Document들을 공백 기준으로 토크나이징

2. shingling(혹은 n-gram)
아래 예시에서는 2-gram으로 shingles를 적용
문서 A : "I love cats. Cats are cute."
- 1번째 2-gram : "I love"
- 2번째 2-gram : "love cats"
- 3번째 2-gram : "cats. Cats"
- 4번째 2-gram : "Cats are"
- 5번째 2-gram : "are cute."

Shingles(A) = { "I love", "love cats.", "cats. Cats", "Cats are", "are cute." }

문서 B : "I really love cats. Cats are so cute."
- 1번째 2-gram : "I really"
- 2번째 2-gram : "really love"
- 3번째 2-gram : "love cats."
- 4번째 2-gram : "cats. Cats"
- 5번째 2-gram : "Cats are"
- 6번째 2-gram : "are so"
- 7번째 2-gram : "so cute."

Shingles(B) = { "I really", "really love", "love cats.", "cats. Cats", "Cats are", "are so", "so cute." }

문서 C : "Dogs are loyal friends"
- 1번째 2-gram : "Dogs are"
- 2번째 2-gram : "are loyal"
- 3번째 2-gram : "loyal friends"

Shingles(C) = { "Dogs are", "are loyal", "loyal friends." }

3. Shingle 집합 정의
Shingles(A) = { "I love", "love cats.", "cats. Cats", "Cats are", "are cute." }
Shingles(B) = { "I really", "really love", "love cats.", "cats. Cats", "Cats are", "are so", "so cute." }
Shingles(C) = { "Dogs are", "are loyal", "loyal friends." }

4. 여러 해시함수 적용 (MinHash Signature 생성)
예를 들어 아래 해시함수 3개(h1, h2, h3)를 정의하여 적용 가능. 실제로는 수십 ~ 수백개를 쓰기도 함
가상의 해시함수를 랜덤하게 정의할 수 있음
  1) h1(x) = (ascii-sum(x) x 13) // 100
  2) h2(x) = (ascii-sum(x) x 7 +3) // 100
  3) h3(x) = CRC32(x) // 100

Documant A의 Shingle들에 해시함수를 적용한 결과 해시값을 Table로 표현하면 아래와 같음

Shingle	h1	h2	h3
"I love"	23	8	17
"love cats."	55	14	33
"cats. Cats"	44	19	45
"Cats are"	11	11	12
"are cute."	92	9	20

이 때, Document A의 MinHash값은, min(h1) = 11, min(h2) = 8, min(h3) = 12가 되므로
MinHash(A) = (11, 8, 12)

Documant B의 Shingle들에 해시함수를 적용한 결과 해시값을 Table로 표현하면 아래와 같음

Shingle	h1	h2	h3
"I really"	62	20	33
"really love"	71	15	29
"love cats."	55	14	33
"cats. Cats"	44	19	45
"Cats are"	11	11	12
"are so"	93	10	21
"so cute."	90	8	22

이 때, Document B의 MinHash값은, min(h1) = 11, min(h2) = 8, min(h3) = 12가 되므로
MinHash(B) = (11, 8, 12)

Documant C의 Shingle들에 해시함수를 적용한 결과 해시값을 Table로 표현하면 아래와 같음

Shingle	h1	h2	h3
"Dogs are"	30	17	52
"are loyal"	87	12	57
"loyal friends."	66	14	36

이 때, Document C의 MinHash값은, min(h1) = 30, min(h2) = 12, min(h3) = 36가 되므로
MinHash(C) = (30, 12, 36)

5. MinHash Signature 비교
MinHash(A) = (11, 8, 12)
MinHash(B) = (11, 8, 12)
MinHash(C) = (30, 12, 36)

1) A vs B
위치별 Signature 비교 : (11 = 11), (8 = 8), (12 = 12)
3개가 모두 동일하므로 Jacaard Index = 3 / 3 = 1
Threshold 0.8보다 높으므로 Near-Dedup으로 판정

2) A vs C, B vs C
위치별 Signature 비교: 모두 일치하지 않으므로 Jacaard Index = 0 / 6 = 0

LSH

고차원 벡터(임베딩 등)이나 집합(Shingle 집합) 간의 유사도를 빠르고 효율적으로 찾기 위한 기법
LSH는 의도적으로 유사한 데이터들은 충돌을 일으켜 같은 버킷에 들어가도록 설계함

6. 버킷 생성
[MinHash Signature] -> (버킷의 Key)로 사용됨
MinHash의 Signature가 생성되면 같은 Signature는 같은 버킷에 담긴다
예를 들어, 2개의 해시함수로 생성된 MinHash Signature의 경우,

- Document A의 Signature = (23, 40)
- Document B의 Signature = (25, 42)
- Document C의 Signature = (70, 95)
- Document D의 Signature = (23, 40)
- Document E의 Signature = (65, 10)

A와 D는 (23, 40)으로 같은 버킷에 담기게 되어 Deduplicate 후보가 된다
또한 B는 (25, 42)로 A, D와 가깝게 되므로 Deduplicate의 후보가 될 수 있다
C, E의 경우는 완전히 다른 Signature로 평가한다

7. 버킷 내 정밀 비교 & Dedup
동일 버킷 / 근접 버킷 안의 Document 끼리만 실제 텍스트를 비교한다 (Distance, Jaccard, Exact Match 등)
A, D의 Shingle의 Jacaard Index를 구하여 Dedup 판단
A, B의 Shingle의 Jacaard Index를 구하여 Dedup 판단
...

A Further Details on NearDup

Document는 먼저 Space를 기준으로 Tokenized됨

그리고 Tabluation Hashing을 사용하여 연속된 5-gram으로 해시됨

이 해시 집합들은 Document의 Signature가 되며 k개의 해시함수를 씀

각 MinHash 값들은 r개의 파티션으로 분할하여 담기며, 각 버킷은 b개의 MinHash값으로 구성됨

만약 두 Document가 적어도 1개 이상의 버킷에서 같은 값을 갖게 되면 잠재적으로 Dedup 후보가 됨

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

두 Document가 Near Dedup될 잠재적 확률은 다음과 같음

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

적어도 1개 이상의 버킷에서 Signature가 겹치면, Jaccard Index를 계산하여 0.8 이상이면 Edit Distance를 추가로 계산함 Edit Distance가 0.8보다 높은 Document Pair는 Dedup 타겟이됨

where $s_{i,j}$ is the Jaccard index between the two documents i and j. 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.

Jaccard Index, Edit Distance를 0.9로도 설정해봄

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 4 shows the histogram of Jaccard similarities and edit similarities for all document pairs which collided in min-hash space, for our chosen configuration (blue) and for the alternative configuration (orange). This allows us verify if the threshold chosen has few comparisons around the chosen threshold, then we've likely captured the majority of actual near duplicates above that threshold. To verify that yourself, look at the left hand tails of the distributions. Since both 0.8 and 0.9 begin to vanish at the same point (in spite of the fact that the two thresholds are optimized for accuracy around different thresholds), we feel comfortable saying that we're capturing the majority of actual near duplicates.

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

문서 갯수 N이고, 한 문서의 최대 토큰 수를 T
Edit Similarity를 최악의 경우 복잡도가 T^2이 됨

왼쪽 항은 Signature를 기준으로 그룹화하는 복잡도

$$O(N + bk^2T^2N) = O(N)$$

오른쪽 항은 모든 문서가 동일한 B개의 버킷에 모여 충돌이 일어나는 병목 상황

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.

recall을 높이기 위해 앞에와 같이 k와 b를 설정함
다른 파라미터를 사용하면 좀더 효율적일 수 있음

B Further Details on ExactSubstr

병렬화된 Linear time Suffix Array 알고리즘을 만듦

SA-IS 알고리즘을 바탕으로 Black-box를 만들었지만 병렬화를 직접 구현하기 어려우므로 Devide and Conquer 접근법을 사용함

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. (2009); Ko and Aluru (2003). Unfortunately, this algorithm is not easily parallelized directly, so we introduce a simple divide and conquer approach to parallelizing the array construction.

Rust로 구현하였고, 기존 알고리즘을 3개 정도 수정함

1) 4GB 이상의 데이터셋 처리하도록 수정

2) UTF-8 대신 Byte Sequence를 처리할 수 있도록 수정

3) Suffix Array를 디스크로부터 스트리밍 할 수 있도록 구현

3번이 중요함

We build our implementation in Rust and extend an existing suffix array library[6] 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.

Divide and conquer는 먼저 데이터셋을 K개로 Split하고 각각의 Chunk를 독립적으로 SA-IS 알고리즘을 적용함

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

경계조건때문에 분할한 Sequence와 Suffix Array A를 붙이는 것은 쉽지 않음

Given two suffix arrays A_1 and A_2 for two sequences S_1 and S_2 it's not completely trivial to construct a single suffix array A for $S = S_1 \mid\mid S_2$ because of the boundary conditions. Instead, we

[6] https://github.com/BurntSushi/suffix

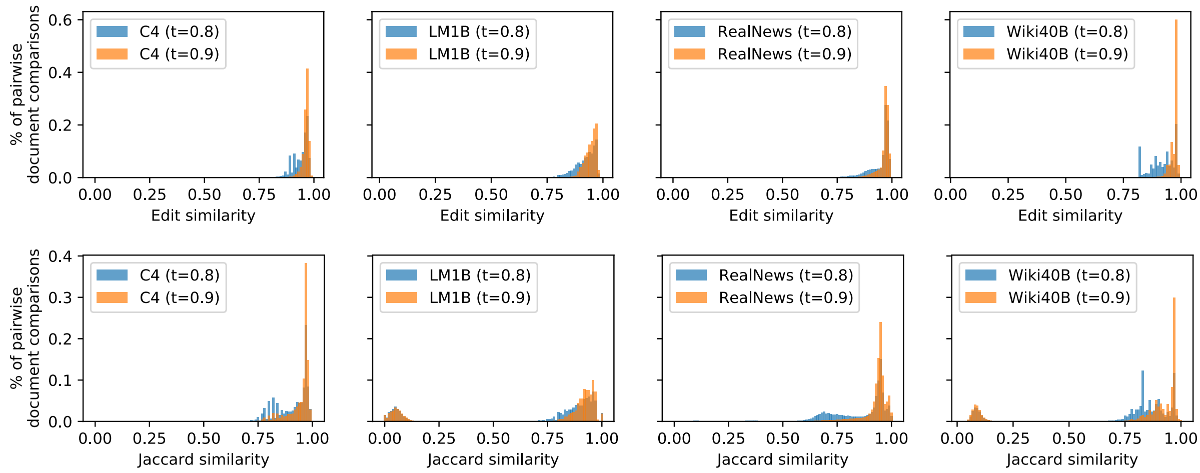


Figure 4: Histograms of document similarities.

예를 들어 K = 4,
S = "ABBANANA"이면,
S1 = "AB"
S2 = "BANANA" 이면, S2[upto4]는 S2의 앞 4개를 가져오므로
S'1 = AB || BANA = ABBANA가 됨

don't build the data $S = S_1 \mid\mid S_2$ but rather let $S_1' = S_1 \mid\mid S_2[uptoK]$ for some K greater than the longest substring match. Then we build the arrays on S_1' and S_2. To merge the arrays together we can remove the items from the first array after index $|S_1|$ and merge-sort insert them into the second.

S1'에서 |S1| 다음 인덱스의 접미사를 제거하고,
제거된 접미사를 S2에 추가함
여기서 S1 = "AB"이면, AB 뒤의 BANA, ANA, NA, A를
S2에 병합정렬시킴

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.

Partial Suffix Array
B와 C를 합친다고 가정

Sbi < Sci 이면
Bi를 Suffix Array A
에 추가하고,
반대의 경우는
Ci를 Suffix Array
A에 추가함

K Split에 일반화
하려면 단순히
Singular하게 비교
하지 말고 Min heap
사용하면됨

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

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

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

Note that this algorithm still requires that the dataset itself fits in memory (so that we can efficiently index in arbitrary positions), but we do not need to fit the entire suffix array into memory. This is fortunate since our suffix array requires an $8\times$ 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.

Setting a threshold of duplicates. An important question is how long must a substring match be before it is counted as a duplicate. In Figure 5, 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

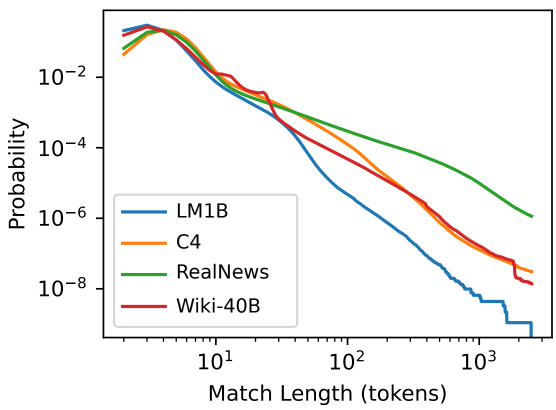


Figure 5: For each substring of length k, we plot the probability that there exists a second identical length-k substring in the same train set. Matches with length under 10 subword tokens are common, and account for 90% of tokens. We choose a threshold of 50 for experiments.

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.

C Further Details on Model Training

Each model was trained for 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 (Shazeer and Stern, 2018). 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.

D Energy Consumption

We trained for approximately 131 hours or 5.5 days on a 128-core TPU v3. The approximate deduplicated dataset is 3.9% smaller than the original dataset and trains in 63 hours/epoch, saving us around 5 hours of compute time for the two epochs. The XL-ORIGINALmodel was trained in North America where the XL-EXACTSUBSTR and XL-NEARDUP were trained in Taiwan. We used data from Patterson et al. (2021) to estimate amount of energy used in training these models by computing the amount of MWh/hour/core and multiplying by our usage (see Table 6 for how we computed these values). For simplicity, we use estimates from Taiwainese datacenters as an estimate. We estimate training 2 epochs of XL-ORIGINAL and XL-EXACTSUBSTR uses $5.86 MWh$. XL-NEARDUP is trained for fewer steps and we estimate uses $5.63 MWh$. Training each base model was approximately 3 days on a 64-core TPU v3 pod slice which uses an estimated $1.61 MWh$.

In addition to model training, evaluation and inference were performed on 64-core TPU v3 pod slices. Generating 100,000 sequences from the XL models takes approximately 0.64 hours. We generated 100,000 sequences for each of five types of prompts for two checkpoints of the model for a total of 1M sequences per model. This took approximately 19.2 hours. We estimate generating 3M sequences uses $0.43 MWh$.

E More Results

Qualitative Examples. Table 8 shows several examples of pairs of documents in C4 whose edit distance is close to our chosen edit similarity threshold of 0.8. Table 9 shows substrings which were identified by EXACTSUBSTR as being in C4 more than once. Table 10 shows several examples of unprompted generations which were identified as memorized are shown.

Distribution of memorization. Figure 6 shows the distribution in memorization amount over all generated sequences when using four types of prompting: train example with duplicates in train,

	T5 11B	XL-Original XL-ExactSubstr	XL-NearDup	Base-Original Base-ExactSubstr	Total Inference
TPU v3 cores	512	128	128	64	64
Training time (days)	20	5.47	5.26	3	0.80
TPU hrs	245760	16804.70	16149.31	4608	1228.80
Energy (MWh)	85.70	5.86	5.63	1.61	0.43

Table 6: Estimates of energy usage based on the data in Patterson et al. (2021). The first column is Patterson et al. (2021)'s estimate of the T5 11B encoder-decoder model, which we based our own estimates on. Inference includes all XL models. We generated 100,000 sequences from 3 models, with 5 prompts, and at 2 different checkpoints.).

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 .
RealNews	KUALA LUMPUR (Reuters) - Roads in Southeast Asia have been getting a little louder lately as motorcycle makers, an aspiring middle class and easy bank credit come together to breed a new genus of motorcyclists – the big-bike rider. [...]	A visitor looks at a Triumph motorcycle on display at the Indonesian International Motor Show in Jakarta September 19, 2014. REUTERS/Darren Whiteside\nKUALA LUMPUR (Reuters) - Roads in Southeast Asia have been getting a little [...] big-bike rider. [...]
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 7: 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.

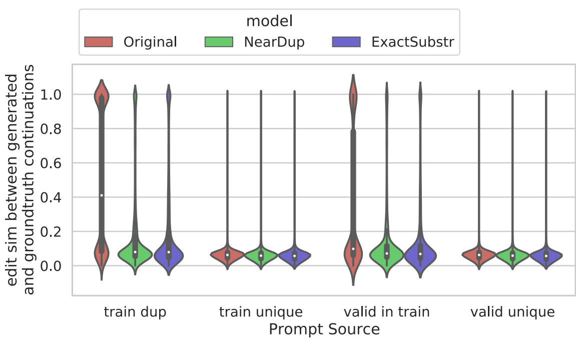


Figure 6: Memorized continuations distribution

train examples without any duplicates, validation examples with duplicates in train, and validation examples without any duplicates.

URLs with many duplicates. Table 11 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.

NearDup cluster sizes. Figure 8 shows the distribution of cluster sizes from running NearDup on RealNews, LM1B, and Wiki-40B (results for C4 are in Figure 1 the main paper).

Dataset Sizes Table 13 gives the size in BPE tokens and in examples of each dataset before and after deduplication. Because most datasets were

Due to high demand, we have yet to critique this request. That said, we assure that the review will be produced in due time by our dilligent and unwavering staff in a professional manner. This site is highly regarded amongst its peers in terms of speed and reliability, so feel free to check us out!	Due to a heavy overflow, we have not been able to critique this request. That said, we assure that the review will be produced in due time by our dilligent and unshakable staff in a professional manner. This site is highly regarded amongst its peers in terms of efficiency and reliability, so feel free to visit!
Need Pop Tacos parking? You can reserve parking near Pop Tacos with SpotHero. Find low rates without parking coupons by booking a guaranteed spot online. Avoid circling, getting ticketed or running out to feed your meter. Search our parking map, compare parking rates and reserve a discounted parking spot today. Happy parking, and enjoy your meal at Pop Tacos!	Il Sole parking. Reserve parking near Il Sole in NYC.\nYou can reserve parking near Il Sole with SpotHero. Find low rates without parking coupons by booking a guaranteed spot online. Avoid circling, getting ticketed or running out to feed your meter. Search our parking map, compare parking rates and reserve a discounted parking spot today. Happy parking, and enjoy your meal at Il Sole!
This item was available on Vinyl 7" but is now sold out on all formats, sorry. Take a look at what else we have in by Jumbo, check out some related artists, head over to our new releases or knock yourself out reading our latest music news & album reviews.\n2nd single edn of 550.	This item was available on CD but is now sold out on all formats, sorry. Take a look at what else we have in by Sirconical, Misty Dixon, Various, check out some related artists, head over to our new releases or knock yourself out reading our latest music news & album reviews.\nTwisted Nerve comp mini album.
Here is all the information you need about "No One Killed Jessica" on American Netflix. Details include the date it was added to Netflix in the USA, any known expiry dates and new episodes/seasons, the ratings and cast etc. So scroll down for more information or share the link on social media to let your friends know what you're watching.	Here is all the information you need about "A Land Imagined" on Netflix in the UK. Details include the date it was added to UK Netflix, any known expiry dates and new episodes/seasons, the ratings and cast etc. So scroll down for more information or share the link on social media to let your friends know what you're watching.
8 + 8 = Solve this simple math problem and enter the result. E.g. for 1+3, enter 4.	Math question * 7 + 1 = Solve this simple math problem and enter the result. E.g. for 1+3, enter 4.
Long Island College Hospital is committed to providing outstanding patient care in the Brooklyn, NY area, but before you commit to Long Island College Hospital for a Endometrial Ablation make sure you compare and shop other medical facilities. It may save you hundreds (in some cases thousands) of dollars. View a Endometrial Ablation cost comparison for Brooklyn and Request a Free Quote before you make a decision.	Morristown Memorial Hospital is committed to providing outstanding patient care in the Morristown, NJ area, but before you commit to Morristown Memorial Hospital for a Breast Ultrasound make sure you compare and shop other medical facilities. It may save you hundreds (in some cases thousands) of dollars. View a Breast Ultrasound cost comparison for Morristown and Request a Free Quote before you make a decision.

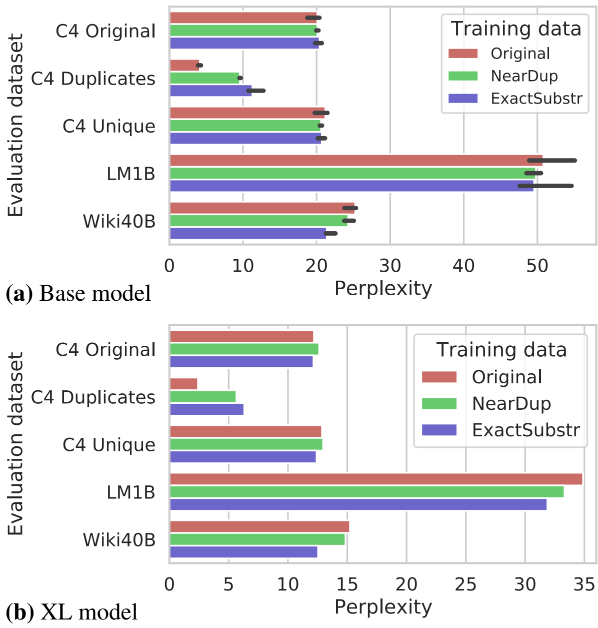
Table 8: Several examples of pairs of documents in C4 that were found by the Approximate Matching algorithm and identified as having edit similarity of almost exactly 0.8. Pairs of documents less similar than 0.8 were not identified as duplicates. For readability, matching subsequences have been highlighted.

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 around 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 9: 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.

already deduplicated of exact matches during their creation, EXACTSUBSTRdeduplication does not actually remove any examples.

Perplexity on LM1B. Figure 7 is the same as Figure 2 of the main paper, except with perplexity on LM1B included. LM1B was omitted from the main paper's figure in order to improve readability.



(a) Base model

(b) XL model

Figure 7: 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).

Generated Text	**Freq in C4**
, you'll need to be knowledgeable to make the very best decisions. We will make sure you know what can be expected. We take the surprises from the picture by giving accurate and thorough information. You can start by talking about your task with our client service staff when you dial 888-353-1299. We'll address all of your questions and arrange the initial meeting. We work closely with you through the whole project, and our team can show up promptly and prepared.	5,497
then Waterside Lodge are well equipped for the task. Our fully equipped family sized lodges offer a comfortable luxurious stay for a fantastic price, giving you beautiful views of the lakes and the surrounding countryside. Offering luxurious self-catering holidays in our fully featured Scandinavian holiday lodges. Perfectly located to explore the beaches, coastline. All of our lodges are sized for 6 people and are furnished to the highest standards to ensure you have a stay like no other. At Waterside Lodge the stay itself is only half of the package, Waterside lodge is situated closely to the Heritage Coast which makes our lodges the perfect stay for anyone wanting to get away and have a relaxing countryside break from the city. Whilst you stay with us be sure to take advantage of all the activities Waterside Lodge has to offer. Such as the use of our on-site fishing lakes for the keen fisherman, free internet access, outside relaxation areas, comfortable lounges and much more.	571
you are only looking to find rent to own homes in your city or are open to exploring all kinds of rent to own home listings, our database does it all. One of the best aspects of iRentToOwn.com is that, besides options to rent to buy a house, it has numerous other categories of home sale options. These include bank foreclosure homes, pre-foreclosure homes, short sales, HUD/government foreclosures, auction homes and owner-financing/FSBO (For Sale By Owner) homes. With help from the convenient search features offered by our site, shoppers are able to find their ideal lease to own home, real estate company, and more in South	51
, IL employs journeyman as licensed to work by themselves, without direct supervision, installing wiring, outlets and fixtures. Our journeyman also does service work, troubleshooting when a breaker fails or a light stops working. Our journeyman does not offer permits that must be issued by our master. Our journeyman follows our master's plans and directions. Our journeyman's responsibilities will vary based on the work that needs to be done. Our journeymen are skilled with residential, commercial and industrial installations and repairs.ust work from six years as an apprentice, under direct supervision of our master, and pass a journeyman test. This person also must have some classroom education on the National Electrical Code and fundamental electricity in a technical school a program affiliated with the National Joint Apprenticeship Training Council. Journeyman training combines hands-on work with education on basic electricity.	6
combustion process of a petrol engine is never perfect. Dangerous gases, such as nitrogen oxide, carbon monoxide and hydrocarbons will arise and it is the job of the catalytic converter to reduce these to safer emissions. These cat converters can fail by becoming clogged, or if the engine has bad exhaust valves or the plugs fail, causing unburned fuel to overheat the converter. Mettam's Mufflers can resolve these issues with your Karr	5
,ANDREW Find the ancestral town: Many a researcher is stuck behind records that say, 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 f their records, too. Around 1900, the Roman Catholic churches reported marriages to the churches where the persons were baptised, and before the wedding, they would require a baptismal certificate from that church, without marriage notations, to make sure that the persons were no t already married, ordained, or whatever, and were free to marry. Do check the Catholic records especially for ex loco and the home town. If your ancestor's sister had a daughter who generated a marriage or death record saying, MOTHER'S BIRTHPLACE: and the exact town, then y ou know where to start searching for records that will confirm it is your ancestor's home town. BEWARE: Just because you find a family with the same names does not mean they are the same family, as they could very well be an unrelated family from a different town in the same an cestral country. The webmaster has learned this. One clue was that one family was still having babies in Potenza city, Italy while the other was having babies in Colorado, U.S.A.	2
will not want to search for Power Washing companies in Wyoming on an extensive basis. The service personnel will be at your doorsteps through online or phone booking. The power wash solutions offered by us are matchless and you can compare with others in Winfield, IL. The power wash services offered by us are very economical. Gutter brightener will be applied which will be followed by cleaning through double scrub. The cleaning will be done by using a soft bristle brush. The bond and contaminants will be released in an effortless manner.	1
Z3 Plus are valid in all major cities of India like Delhi, Gurgaon, Noida, Mumbai, Chennai, Bangalore, Hyderabad, Kolkata, Pune, Ahmedabad, Coimbatore, Lucknow, Trichy, Madurai, Trivandrum, Mysore, Jaipur, Chandigarh, Pondicherry, Bhopal, Patna, Bhubaneswar, Amritsar, Cochin, Allahabad, Srinagar, New Delhi, Surat, Ludhiana, Navi Mumbai, Ghaziabad, Bengaluru, Indore, Nagpur, Thane, Agra, Meerut, Ranchi. The delivery feasibility and charges may be varying, hence for them please check with the particular seller or store.	1

Table 10: A selection of substrings generated by XL-ORIGINAL with no prompting (and top-k with k=50) that were identified by EXACTSUBSTR as being in C4 multiple times. The number of times each substring was found in C4 is given. We observe that most memorized generations tend to be from advertisements.

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

Table 11: 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.

Training Dataset:	C4-ORIGINAL		C4-NEARDUP		C4-EXACTSUBSTR	
Epoch:	1	2	1	2	1	2
No prompt	1.93%	1.57%	0.19%	0.26%	0.14%	0.17%
Duplicate Train Prompts	35.88%	34.34%	3.34%	3.15%	5.71%	4.67%
Unique Train Prompt	0.42%	0.41%	0.42%	0.41%	0.22%	0.23%
Duplicate Test Prompt	16.27%	15.32%	1.61%	1.52%	0.34%	0.25%
Unique Test Prompt	0.25%	0.22%	0.21%	0.23%	0.03%	0.08%

Table 12: Percentage of tokens in 100k generations that were part of memorized substring according to EXACT-SUBSTR. Models trained with approximate or exact deduplication have $10\times$ less memorization than the model trained on the original (non-deduplicated) dataset.

	Final train set size in tokens			Final train set size in examples		
	ORIGINAL	NEARDUP	EXACTSUBSTR	ORIGINAL	NEARDUP	EXACTSUBSTR
C4	177.3B	173.7B	165.4B	364.87M	350.48M	350.48M
Real News	24.7B	22.4B	20.1B	31.16M	28.39M	28.39M
LM1B	1.0B	0.94B	0.90B	30.30M	29.87M	30.16M
Wiki40B	2.25B	2.24B	2.19B	2.93M	2.91M	2.93M

Table 13: Each row shows the size in tokens (according to our 50k BPE vocab) and in examples of a train set in its original form, with NEARDUP deduplication, and with EXACTSUBSTR deduplication.

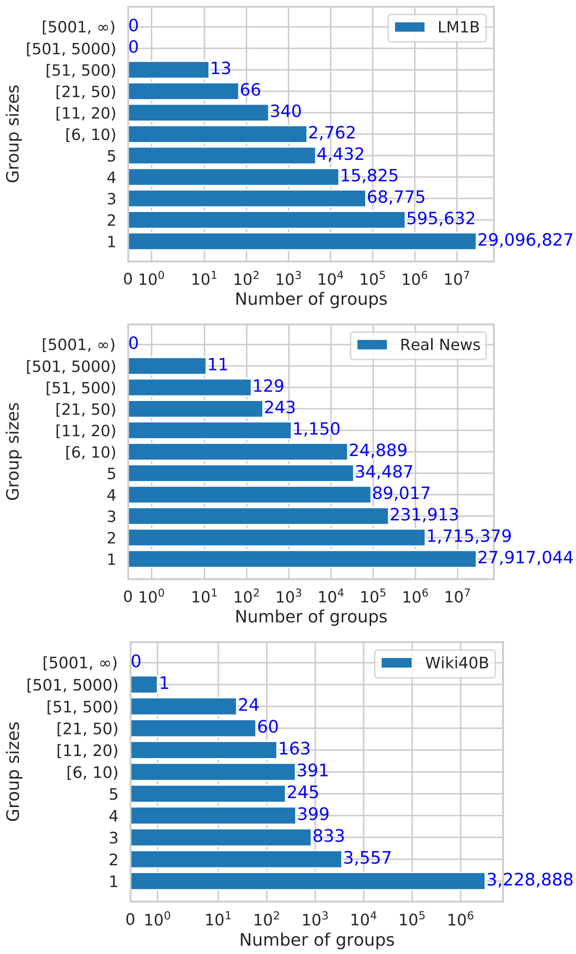


Figure 8: The distribution of near-duplicate cluster sizes from running NEARDUP on each dataset.