Deduplicating Training Data Makes Language Models Better[1]

CS 592-LLM – Adv Topics in Reasoning with Large Language Models

Presented by J. Setpal

November 2, 2023‘

[1] Lee, Katherine, et al. (ACL 2022)

Outline

1. Task Overview
2. Methodology
3. Results
4. Discussion

Outline

1. Task Overview
2. Methodology
3. Results
4. Discussion

Let's Talk Datasets

Large Language Models are characterized by large #parameters.

Let's Talk Datasets

Large Language Models are characterized by large #parameters.

This leaves them vulnerable to *memorization*.

Let's Talk Datasets

Large Language Models are characterized by large #parameters.

This leaves them vulnerable to *memorization*. A key factor in promoting *generalization* has been the introduction of **large datasets**.

Let's Talk Datasets

Large Language Models are characterized by large #parameters.

This leaves them vulnerable to *memorization*. A key factor in promoting *generalization* has been the introduction of **large datasets**.

As a consequence, manual review and curation is *expensive*, and larger datasets suffer in quality.

It is a consequence of **lack of due diligence**.

Contamination by Duplication

One pervasive source of dataset bias is duplicated training samples.

Contamination by Duplication

One pervasive source of dataset bias is duplicated training samples.

Sometimes, **validation sets contain training samples**.

Contamination by Duplication

One pervasive source of dataset bias is duplicated training samples.

Sometimes, **validation sets contain training samples**.

This *promotes* memorization because:

1. Repeat samples are upweighted during training.

Contamination by Duplication

One pervasive source of dataset bias is duplicated training samples.

Sometimes, **validation sets contain training samples**.

This *promotes* memorization because:

1. Repeat samples are upweighted during training.
2. Validation scores are highest when duplicated data is memorized.

Contamination by Duplication

One pervasive source of dataset bias is duplicated training samples.

Sometimes, **validation sets contain training samples**.

This *promotes* memorization because:

1. Repeat samples are upweighted during training.
2. Validation scores are highest when duplicated data is memorized.
3. Regularizers promote high scores with fewer parameters.

Contamination by Duplication

One pervasive source of dataset bias is duplicated training samples.

Sometimes, **validation sets contain training samples**.

This *promotes* memorization because:

1. Repeat samples are upweighted during training.
2. Validation scores are highest when duplicated data is memorized.
3. Regularizers promote high scores with fewer parameters.

This paper improves generalization performance by **deduplicating data samples**.

Advantages to Deduplication

More concretely:

1. $> 1\%$ of tokens emitted unprompted from a biased model are part of a **memorized sequence**.

Advantages to Deduplication

More concretely:

1. $> 1\%$ of tokens emitted unprompted from a biased model are part of a **memorized sequence**. Deduplication reduced this to $\approx 0.1\%$.

Advantages to Deduplication

More concretely:

1. $> 1\%$ of tokens emitted unprompted from a biased model are part of a **memorized sequence**. Deduplication reduced this to $\approx 0.1\%$.
2. A *61-word sequence* was found to repeat $\mathbf{61,036}$ times in training and $\mathbf{61}$ times in validation in the C4 dataset.

Advantages to Deduplication

More concretely:

1. $> 1\%$ of tokens emitted unprompted from a biased model are part of a **memorized sequence**. Deduplication reduced this to $\approx 0.1\%$.
2. A *61-word sequence* was found to repeat $\mathbf{61,036}$ times in training and $\mathbf{61}$ times in validation in the C4 dataset.
3. Training models on deduplicated datasets improves training efficiency. Deduplicated datasets are upto 19% smaller!

Advantages to Deduplication

More concretely:

1. $> 1\%$ of tokens emitted unprompted from a biased model are part of a **memorized sequence**. Deduplication reduced this to $\approx 0.1\%$.
2. A *61-word sequence* was found to repeat $\mathbf{61,036}$ times in training and $\mathbf{61}$ times in validation in the C4 dataset.
3. Training models on deduplicated datasets improves training efficiency. Deduplicated datasets are upto 19% smaller!
4. Deduplication does not hurt *perplexity*; in cases it reduces perplexity by upto 10%.

Advantages to Deduplication

More concretely:

1. $> 1\%$ of tokens emitted unprompted from a biased model are part of a **memorized sequence**. Deduplication reduced this to $\approx 0.1\%$.
2. A *61-word sequence* was found to repeat $\mathbf{61,036}$ times in training and $\mathbf{61}$ times in validation in the C4 dataset.
3. Training models on deduplicated datasets improves training efficiency. Deduplicated datasets are upto 19% smaller!
4. Deduplication does not hurt *perplexity*; in cases it reduces perplexity by upto 10%. Deduplication also improves rate of convergence.

Dataset Evaluations #1

The authors analyze deduplication in four datasets of varying sizes.

Dataset Evaluations #1

The authors analyze deduplication in four datasets of varying sizes.

Wikipedia (Wiki-40B) consists of $2.9M$ pages of cleaned wikipedia text at avg. $768BPE$. Besides *redirects*, no data deduplication was carried out.

Dataset Evaluations #1

The authors analyze deduplication in four datasets of varying sizes.

Wikipedia (Wiki-40B) consists of $2.9M$ pages of cleaned wikipedia text at avg. $768BPE$. Besides *redirects*, no data deduplication was carried out.

One-Billion word Benchmark (LM1B) contains $30M$ single-sentence samples of news commentary. It has a **13.2%** train-test overlap.

Dataset Evaluations #1

The authors analyze deduplication in four datasets of varying sizes.

Wikipedia (Wiki-40B) consists of $2.9M$ pages of cleaned wikipedia text at avg. $768BPE$. Besides *redirects*, no data deduplication was carried out.

One-Billion word Benchmark (LM1B) contains $30M$ single-sentence samples of news commentary. It has a **13.2%** train-test overlap.

Note

The authors limit the scope of their research to english-only subsets.

Dataset Evaluations #2

Colossal Cleaned Common Crawl (C4) contains $360M$ documents at an avg. $486BPE$.

Dataset Evaluations #2

Colossal Cleaned Common Crawl (C4) contains $360M$ documents at an avg. $486BPE$. It follows a more complicated cleaning process:

1. Each is paragraph is **hashed**, and <u>hash collisions are excluded</u>.

Dataset Evaluations #2

Colossal Cleaned Common Crawl (C4) contains $360M$ documents at an avg. $486BPE$. It follows a more complicated cleaning process:

1. Each is paragraph is **hashed**, and hash collisions are excluded.
2. *Placeholder text, code* and *prohibited words* are removed.

Dataset Evaluations #2

Colossal Cleaned Common Crawl (C4) contains $360M$ documents at an avg. $486BPE$. It follows a more complicated cleaning process:

1. Each is paragraph is **hashed**, and hash collisions are excluded.
2. *Placeholder text, code* and *prohibited words* are removed.

Real News is a subset of Common Crawl, containing $31M$ documents at an avg. $793BPE$.

Dataset Evaluations #2

Colossal Cleaned Common Crawl (C4) contains $360M$ documents at an avg. $486BPE$. It follows a more complicated cleaning process:

1. Each is paragraph is **hashed**, and hash collisions are excluded.
2. *Placeholder text, code* and *prohibited words* are removed.

Real News is a subset of Common Crawl, containing $31M$ documents at an avg. $793BPE$. It's deduplicated by passing the first 100 tokens against a **bloom filter**. Documents containing hash collisions are excluded.

Dataset Evaluations #2

Colossal Cleaned Common Crawl (C4) contains $360M$ documents at an avg. $486BPE$. It follows a more complicated cleaning process:

1. Each is paragraph is **hashed**, and hash collisions are excluded.
2. *Placeholder text, code* and *prohibited words* are removed.

Real News is a subset of Common Crawl, containing $31M$ documents at an avg. $793BPE$. It's deduplicated by passing the first 100 tokens against a **bloom filter**. Documents containing hash collisions are excluded.

However, these deduplication strategies are simply *not good enough*.

Outline

Exact Substring Deduplication

Consider $D := \{x_{\boldsymbol{i}}\}_{\boldsymbol{i=1}}^{N}$ where x_i are dataset samples such that $x_i := [x_i^{\boldsymbol{h}}]_{\boldsymbol{h=1}}^{|S|}$ is the series of tokens comprising the sample.

Exact Substring Deduplication

Consider $D := \{x_{\boldsymbol{i}}\}_{\boldsymbol{i=1}}^{N}$ where x_i are dataset samples such that $x_i := [x_i^{\boldsymbol{h}}]_{\boldsymbol{h=1}}^{|S|}$ is the series of tokens comprising the sample.

Idea: It's rare for a sequence of words to be recreated verbatim without originating from a shared source.

Exact Substring Deduplication

Consider $D := \{x_{\boldsymbol{i}}\}_{\boldsymbol{i=1}}^{N}$ where x_i are dataset samples such that $x_i := [x_i^{\boldsymbol{h}}]_{\boldsymbol{h=1}}^{|S|}$ is the series of tokens comprising the sample.

Idea: It's rare for a sequence of words to be recreated verbatim without originating from a shared source.

Therefore it is important to also <u>deduplicate substring matches</u>.

Exact Substring Deduplication

Consider $D := \{x_{\boldsymbol{i}}\}_{\boldsymbol{i=1}}^{N}$ where x_i are dataset samples such that $x_i := [x_i^{\boldsymbol{h}}]_{\boldsymbol{h=1}}^{|S|}$ is the series of tokens comprising the sample.

Idea: It's rare for a sequence of words to be recreated verbatim without originating from a shared source.

Therefore it is important to also <u>deduplicate substring matches</u>.

When samples $\{x_i, x_j\}$ exist such that $[x_i^h]_{h=a}^k = [x_j^h]_{h=b}^l; k - a = l - b$, there is an exact substring match and must be deduplicated.

Exact Substring Deduplication

Consider $D := \{x_{\boldsymbol{i}}\}_{\boldsymbol{i=1}}^{N}$ where x_i are dataset samples such that $x_i := [x_i^{\boldsymbol{h}}]_{\boldsymbol{h=1}}^{|S|}$ is the series of tokens comprising the sample.

Idea: It's rare for a sequence of words to be recreated verbatim without originating from a shared source.

Therefore it is important to also deduplicate substring matches.

When samples $\{x_i, x_j\}$ exist such that $[x_i^h]_{h=a}^{k} = [x_j^h]_{h=b}^{l}; k - a = l - b$, there is an exact substring match and must be deduplicated.

Substring length threshold $d^*_{k-a,\ l-b} \geq 50$ is a hyperparameter.

Exact Substring Deduplication

Consider $D := \{x_{\boldsymbol{i}}\}_{\boldsymbol{i=1}}^{N}$ where x_i are dataset samples such that $x_i := [x_i^{\boldsymbol{h}}]_{\boldsymbol{h=1}}^{|S|}$ is the series of tokens comprising the sample.

Idea: It's rare for a sequence of words to be recreated verbatim without originating from a shared source.

Therefore it is important to also <u>deduplicate substring matches</u>.

When samples $\{x_i, x_j\}$ exist such that $[x_i^h]_{h=a}^{k} = [x_j^h]_{h=b}^{l}$; $k - a = l - b$, there is an exact substring match and must be deduplicated.

Substring length threshold $d^*_{k-a,\ l-b} \geq 50$ is a hyperparameter.

When all criterion are met, one substring is excluded, deduplicating the dataset. This approach is called ExactSubstr.

Suffix Arrays

Despite being conceptually simple, EXACTSUBSTR's naive implementation runs in **quadratic time**.

Suffix Arrays

Despite being conceptually simple, ExactSubstr's naive implementation runs in **quadratic time**. It's *significantly* costlier for larger datasets.

Suffix Arrays

Despite being conceptually simple, ExactSubstr's naive implementation runs in **quadratic time**. It's *significantly* costlier for larger datasets.

To improve this, we concate samples $\{x_i\}_{i=1}^{N}$ into a single sequence $\mathcal{S}$.

Suffix Arrays

Despite being conceptually simple, ExactSubstr's naive implementation runs in **quadratic time**. It's *significantly* costlier for larger datasets.

To improve this, we concate samples $\{x_i\}_{i=1}^{N}$ into a single sequence $\mathcal{S}$. We further construct a **Suffix Array** $\mathcal{A}$ of $\mathcal{S}$.

Suffix Arrays

Despite being conceptually simple, EXACTSUBSTR's naive implementation runs in **quadratic time**. It's *significantly* costlier for larger datasets.

To improve this, we concate samples $\{x_i\}_{i=1}^{N}$ into a single sequence $\mathcal{S}$. We further construct a **Suffix Array** $\mathcal{A}$ of $\mathcal{S}$.

A **Suffix Array** is the sorted list of suffixes in a sequence s. Here, $s = \mathcal{S}$.

Suffix Arrays

Despite being conceptually simple, $\textsc{ExactSubstr}$'s naive implementation runs in **quadratic time**. It's *significantly* costlier for larger datasets.

To improve this, we concate samples $\{x_i\}_{i=1}^{N}$ into a single sequence $\mathcal{S}$. We further construct a **Suffix Array** $\mathcal{A}$ of $\mathcal{S}$.

A **Suffix Array** is the sorted list of suffixes in a sequence s. Here, $s = \mathcal{S}$. For instance, if $s =$ "banana":

$$\begin{aligned}\mathcal{A} &= \text{sorted}(\{\text{``banana''}, \text{``anana''}, \text{``nana''}, \text{``ana''}, \text{``na''}, \text{``a''}\}) \\ &= [\text{``a''}, \text{``ana''}, \text{``anana''}, \text{``banana''}, \text{``na''}, \text{``nana''}]\end{aligned}$$

Suffix Arrays

Despite being conceptually simple, $\textsc{ExactSubstr}$'s naive implementation runs in **quadratic time**. It's *significantly* costlier for larger datasets.

To improve this, we concate samples $\{x_i\}_{i=1}^{N}$ into a single sequence $\mathcal{S}$. We further construct a **Suffix Array** $\mathcal{A}$ of $\mathcal{S}$.

A **Suffix Array** is the sorted list of suffixes in a sequence s. Here, $s = \mathcal{S}$. For instance, if $s =$ "banana":

$$\begin{aligned} \mathcal{A} &= \text{sorted}(\{\text{"banana"}, \text{"anana"}, \text{"nana"}, \text{"ana"}, \text{"na"}, \text{"a"}\}) \\ &= [\text{"a"}, \text{"ana"}, \text{"anana"}, \text{"banana"}, \text{"na"}, \text{"nana"}] \end{aligned}$$

$\mathcal{A}(\mathcal{S})$ can be constructed in linear time $O(|\mathcal{S}|)$, and is therefore efficient.

Substring Matching & Parallelism

We can use $\mathcal{A}(\mathcal{S})$ to identify duplicate substrings within the dataset.

Substring Matching & Parallelism

We can use $\mathcal{A}(\mathcal{S})$ to identify duplicate substrings within the dataset.

If sub-sequence s is repeated within $\mathcal{S}$ at positions $\{i, j\}$, $\mathcal{A}_{\mathcal{S}_i} = \mathcal{A}_{\mathcal{S}_{j\pm1}}$.
This is because $\mathcal{A}$ is a **sorted array**.

Substring Matching & Parallelism

We can use $\mathcal{A}(\mathcal{S})$ to identify duplicate substrings within the dataset.

If sub-sequence s is repeated within $\mathcal{S}$ at positions $\{i, j\}$, $\mathcal{A}_{\mathcal{S}_i} = \mathcal{A}_{\mathcal{S}_{j\pm1}}$. This is because $\mathcal{A}$ is a **sorted array**.

Identifying suffixes therefore involves the parallelizable task of searching through $\mathcal{A}$.

The Need for Approximate Matching

Consider the following cases:

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!

The Need for Approximate Matching

Consider the following cases:

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!

Despite significant overlap, duplication is <u>not identified</u> by ExactSubstr.

The Need for Approximate Matching

Consider the following cases:

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!

Despite significant overlap, duplication is <u>not identified</u> by ExactSubstr.

The authors introduce the NearDup algorithm to resolve this.

Let's talk about it!

The NearDup Algorithm #1

Idea: Approximate the *Jaccard Similarity Coefficient* and an *Edit Similarity Score* between two documents $\{x_i, x_j\}$.

The NEARDUP Algorithm #1

Idea: Approximate the *Jaccard Similarity Coefficient* and an *Edit Similarity Score* between two documents $\{x_i, x_j\}$. Approximate duplications exist for *high Jaccard Coefficients* and *high Edit Similarities*.

[2] doi:10.1109/SEQUEN.1997.666900

The NEARDUP Algorithm #1

Idea: Approximate the *Jaccard Similarity Coefficient* and an *Edit Similarity Score* between two documents $\{x_i, x_j\}$. Approximate duplications exist for *high Jaccard Coefficients* and *high Edit Similarities*.

This derives from MINHASH[2]. $J(A, B) \in [0, 1]$.

[2] `doi:10.1109/SEQUEN.1997.666900`

The NearDup Algorithm #1

Idea: Approximate the *Jaccard Similarity Coefficient* and an *Edit Similarity Score* between two documents $\{x_i, x_j\}$. Approximate duplications exist for *high Jaccard Coefficients* and *high Edit Similarities*.

This derives from MinHash[2]. $J(A, B) \in [0, 1]$. Each document is represented by a hash h; in this case the set of n-grams. Only the k-smallest n-grams are used to compute the Jaccard:

$$J(d_{x_i}, d_{x_j}) = \frac{d_{x_i} \cap d_{x_j}}{d_{x_i} \cup d_{x_j}}$$

Here, $h =$ tabulation hashing, $n = 5$ and $k =$.

[2] doi:10.1109/SEQUEN.1997.666900

The NearDup Algorithm #2

Tabulation Hashing is a bucketized hashing algorithm.

The NearDup Algorithm #2

Tabulation Hashing is a bucketized hashing algorithm. Subsequence hashes are computed against each bucket, and a final hash for the document is obtained using bitwise XOR.

The NearDup Algorithm #2

Tabulation Hashing is a bucketized hashing algorithm. Subsequence hashes are computed against each bucket, and a final hash for the document is obtained using bitwise XOR.

The probability score of a match is obtained using the following procedure:

1. The set of hashes obtained per gram is the *document signature*.

The NearDup Algorithm #2

Tabulation Hashing is a bucketized hashing algorithm. Subsequence hashes are computed against each bucket, and a final hash for the document is obtained using bitwise XOR.

The probability score of a match is obtained using the following procedure:

1. The set of hashes obtained per gram is the *document signature*.
2. Each element is hashed using k other hashing functions.

The NearDup Algorithm #2

Tabulation Hashing is a bucketized hashing algorithm. Subsequence hashes are computed against each bucket, and a final hash for the document is obtained using bitwise XOR.

The probability score of a match is obtained using the following procedure:

1. The set of hashes obtained per gram is the *document signature*.
2. Each element is hashed using k other hashing functions.
3. The minimum hashed element for each k function is stored.

The NEARDUP Algorithm #2

Tabulation Hashing is a bucketized hashing algorithm. Subsequence hashes are computed against each bucket, and a final hash for the document is obtained using bitwise XOR.

The probability score of a match is obtained using the following procedure:

1. The set of hashes obtained per gram is the *document signature*.
2. Each element is hashed using k other hashing functions.
3. The minimum hashed element for each k function is stored.
4. They are partitioned into r buckets, with b hashes per bucket.

The NearDup Algorithm #2

Tabulation Hashing is a bucketized hashing algorithm. Subsequence hashes are computed against each bucket, and a final hash for the document is obtained using bitwise XOR.

The probability score of a match is obtained using the following procedure:

1. The set of hashes obtained per gram is the *document signature*.
2. Each element is hashed using k other hashing functions.
3. The minimum hashed element for each k function is stored.
4. They are partitioned into r buckets, with b hashes per bucket.
5. If $\{x_i, x_j\}$ share hashes in ≥ 1 bucket, it is considered a match.

The NearDup Algorithm #2

Tabulation Hashing is a bucketized hashing algorithm. Subsequence hashes are computed against each bucket, and a final hash for the document is obtained using bitwise XOR.

The probability score of a match is obtained using the following procedure:

1. The set of hashes obtained per gram is the *document signature*.
2. Each element is hashed using k other hashing functions.
3. The minimum hashed element for each k function is stored.
4. They are partitioned into r buckets, with b hashes per bucket.
5. If $\{x_i, x_j\}$ share hashes in ≥ 1 bucket, it is considered a match.

Finally, we obtain the following **probability score**:

$$P(d_{x_i}, d_{x_j} | J(d_{x_i}, d_{x_j})) = 1 - (1 - J(d_{x_i}, d_{x_j})^b)^r$$

Here, $b = 20, r = 450$ and $k = br = 9000$

The NearDup Algorithm #3

For document pairs $\{x_i, x_j\}$ considered potential matches, the *full Jaccard Index* is computed.

The NearDup Algorithm #3

For document pairs $\{x_i, x_j\}$ considered potential matches, the *full Jaccard Index* is computed. If $J_F(d_{x_i}, d_{x_j}) \geq 0.8$, the *edit similarity* is computed:

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

The NEARDUP Algorithm #3

For document pairs $\{x_i, x_j\}$ considered potential matches, the *full Jaccard Index* is computed. If $J_F(d_{x_i}, d_{x_j}) \geq 0.8$, the *edit similarity* is computed:

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

Finally, a graph is created to cluster similar documents.

The NearDup Algorithm #3

For document pairs $\{x_i, x_j\}$ considered potential matches, the *full Jaccard Index* is computed. If $J_F(d_{x_i}, d_{x_j}) \geq 0.8$, the *edit similarity* is computed:

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

Finally, a graph is created to cluster similar documents. If documents are considered a match, edges are constructed between the pair.

The NearDup Algorithm #3

For document pairs $\{x_i, x_j\}$ considered potential matches, the *full Jaccard Index* is computed. If $J_F(d_{x_i}, d_{x_j}) \geq 0.8$, the *edit similarity* is computed:

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

Finally, a graph is created to <u>cluster similar documents</u>. If documents are considered a match, edges are constructed between the pair. The <u>connected components form clusters</u>.

The NEARDUP Algorithm #3

For document pairs $\{x_i, x_j\}$ considered potential matches, the *full Jaccard Index* is computed. If $J_F(d_{x_i}, d_{x_j}) \geq 0.8$, the *edit similarity* is computed:

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

Finally, a graph is created to cluster similar documents. If documents are considered a match, edges are constructed between the pair. The connected components form clusters.

Deduplication is performed on these clusters, and a filtered dataset is obtained.

Outline

Amount of Text Deduplicated #1

Both deduplication techniques were run on the 4 mentioned datasets.

Amount of Text Deduplicated #1

Both deduplication techniques were run on the 4 mentioned datasets. In case of *duplication between splits*, training duplications were removed.

Amount of Text Deduplicated #1

Both deduplication techniques were run on the 4 mentioned datasets. In case of *duplication between splits*, training duplications were removed.

	% train tokens with dup in train	% train tokens with dup in valid	% valid with 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: Deduplications made by ExactSubstr

Amount of Text Deduplicated #2

	% train examples with dup in train	% train examples with dup in valid	% valid with 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: Deduplications made by NearDup

Amount of Text Deduplicated #2

	% train examples with dup in train	% train examples with dup in valid	% valid with 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: Deduplications made by NearDup

On average, ExactSubstr removed more content than NearDup, with a notable exception being **LM1B**: it contains shorter token lengths than the ExactSubstr threshold.

Amount of Text Deduplicated #2

	% train examples with dup in train	% train examples with dup in valid	% valid with 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: Deduplications made by NearDup

On average, ExactSubstr removed more content than NearDup, with a notable exception being **LM1B**: it contains shorter token lengths than the ExactSubstr threshold.

Both ExactSubstr and NearDup remove similar content: 77% of training samples NearDup removed from **C4** contained a 50-length match in ExactSubstr.

Impact on Trained Models

Both methods successfully identify deduplication, promoting parameters biased towards memorization.

Impact on Trained Models

Both methods successfully identify deduplication, promoting parameters biased towards memorization. We observe the following comparisons:

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

Impact on Trained Models

Both methods successfully identify deduplication, promoting parameters biased towards memorization. We observe the following comparisons:

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

In addition, existing models models also suffer from this:

1. 1.38% of 25k-GROVER-Mega outputs contained verbatim RealNews matches.

Impact on Trained Models

Both methods successfully identify deduplication, promoting parameters biased towards memorization. We observe the following comparisons:

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

In addition, existing models models also suffer from this:

1. 1.38% of 25k-GROVER-Mega outputs contained verbatim RealNews matches.
2. $> 5\%$ of tokens in the $\approx 200k$ sequences output by GPT-Neo 1.3B contained verbatim Pile[3] matches.

[3] training dataset

Impact on Prompting

The impact of the outputs produced also depends on whether or not a prompt is supplied to the language model.

Impact on Prompting

The impact of the outputs produced also depends on whether or not a prompt is supplied to the language model.

Without prompting, Transformer-XL returned $> 1\%$ of tokens belonging to memorized sub-sequences. With $\textsc{ExactSubstr}$ and $\textsc{NearDup}$, this reduced to $\approx 0.01\%$.

Impact on Prompting

The impact of the outputs produced also depends on whether or not a prompt is supplied to the language model.

Without prompting, Transformer-XL returned $> 1\%$ of tokens belonging to memorized sub-sequences. With $\textsc{ExactSubstr}$ and $\textsc{NearDup}$, this reduced to $\approx 0.01\%$.

With prompting, impacts on model output are *less meaningful*.

Impact on Prompting

The impact of the outputs produced also depends on whether or not a prompt is supplied to the language model.

Without prompting, Transformer-XL returned $> 1\%$ of tokens belonging to memorized sub-sequences. With $\textsc{ExactSubstr}$ and $\textsc{NearDup}$, this reduced to $\approx 0.01\%$.

With prompting, impacts on model output are *less meaningful*. When the prompt includes duplicate samples in the test set, original Transformer-XL returns the groundtruth continuation 40% of the time.

Impact on Prompting

The impact of the outputs produced also depends on whether or not a prompt is supplied to the language model.

Without prompting, Transformer-XL returned $> 1\%$ of tokens belonging to memorized sub-sequences. With $\textsc{ExactSubstr}$ and $\textsc{NearDup}$, this reduced to $\approx 0.01\%$.

With prompting, impacts on model output are *less meaningful*. When the prompt includes duplicate samples in the test set, original Transformer-XL returns the groundtruth continuation 40% of the time.

Even with $\textsc{ExactSubstr}$ and $\textsc{NearDup}$, the groundtruth is *copied* more when prompts original from duplicate samples, rather than unique ones.

Impact on Prompting

The impact of the outputs produced also depends on whether or not a prompt is supplied to the language model.

Without prompting, Transformer-XL returned $> 1\%$ of tokens belonging to memorized sub-sequences. With $\textsc{ExactSubstr}$ and $\textsc{NearDup}$, this reduced to $\approx 0.01\%$.

With prompting, impacts on model output are *less meaningful*. When the prompt includes duplicate samples in the test set, original Transformer-XL returns the groundtruth continuation 40% of the time.

Even with $\textsc{ExactSubstr}$ and $\textsc{NearDup}$, the groundtruth is *copied* more when prompts original from duplicate samples, rather than unique ones.

Further research is required to entirely eliminate memorization tendencies.

Impact on Perplexity

The authors computer the perplexity of trained models on the validation sets.

Impact on Perplexity

The authors computer the perplexity of trained models on the validation sets.

All models were observed to have similar perplexity on **unique** C4 validation samples.

Impact on Perplexity

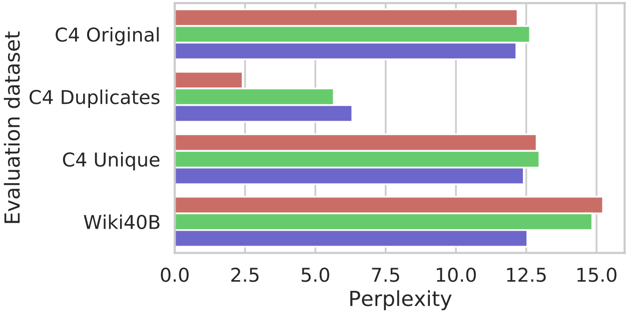
The authors computer the perplexity of trained models on the validation sets.

All models were observed to have similar perplexity on **unique** C4 validation samples. On validation samples *with duplicates*, both approaches have higher perplexity. ExactSubstr has a higher perplexity than NearDup.

Impact on Perplexity

The authors computer the perplexity of trained models on the validation sets.

All models were observed to have similar perplexity on **unique** C4 validation samples. On validation samples *with duplicates*, both approaches have higher perplexity. EXACTSUBSTR has a higher perplexity than NEARDUP.



Outline

Brightspace Questions

1. What are some reasons where data duplication (ergo memorization) is actually useful?
2. Would sentence-vectorization based clustering be a good replacement for NearDup? Why or why not?

Live Disucssion Question

Q: Despite removing a large portion of duplicates, LLMs still suffer from memorization, but *only when prompted with duplicates*. Theorize approaches to solve this.

Thank you!

Have an awesome rest of your day!

Slides: `https://cs.purdue.edu/homes/jsetpal/slides/dedup-td.pdf`