Fast and Scalable Text Deduplication using Locality-Sensitive Hashing

Mar. 2025

1 Background

Large-scale datasets often contain slightly modified, paraphrased, or re-ordered copies of the same content — especially in web crawls, user-generated content, and pretraining corpora for language models. These near-duplicates can distort model training signals, lead to overfitting by inflating the presence of certain linguistic patterns, and may potentially cause data leakage across splits (i.e., overlapping content between training and validation/test sets). Recent research shows that deduplicating training data can significantly improve language model quality [1].

On the other hand, exact pairwise duplicate comparison is computationally infeasible in large-scale settings, as the time complexity is $O(n^2)$, and the redundancy is often fuzzy rather than literal.

2 Objective

This project focuses on building a fast and scalable text deduplication system using Locality-Sensitive Hashing (LSH) techniques. These methods enable efficient detection of near-duplicate documents without relying on expensive pairwise comparisons, making them well-suited for large-scale text preprocessing pipelines.

Students will explore both the theoretical foundations and practical implementations of fingerprinting methods such as MinHash, SimHash, and Bit Sampling, and — most importantly — how these signatures are used in

LSH-based indexing and retrieval mechanisms (e.g., banding, bucketing, or bit sampling-based filtering) to accelerate similarity detection.

A key learning objective is understanding that computing hash signatures alone is not sufficient. Students are expected to implement the entire LSH workflow, including hashing, grouping, and efficient candidate generation — avoiding brute-force pairwise comparisons. This project encourages students to apply LSH principles in a real-world deduplication setting, analyze performance trade-offs, and reflect on the scalability and effectiveness of their designs.

3 Task Description

Each group will implement a modular system that:

- Preprocesses raw text and converts it into suitable feature representations, such as n-grams, token sets, or binary vectors.
- Implements multiple types of fingerprinting methods, including: **MinHash**, **SimHash**, and **Bit Sampling**. These methods should be used to generate compact signatures that approximate text similarity.
- Applies Locality-Sensitive Hashing (LSH) techniques to efficiently identify candidate near-duplicate document pairs. Specifically, students must implement LSH-based filtering strategies such as **banding (for MinHash)**, or other hash-based indexing schemes. **Projects that only compute signatures but perform exhaustive pairwise comparisons will be considered incomplete.**
- Evaluates the system on a real-world corpus: the **test and validation subsets from the Wiki40B English dataset** [1]. The system should detect (1) near-duplicates between the validation and test sets, and (2) near-duplicates within each individual subset.
- Analyzes results both quantitatively (e.g., near-duplicate rate, runtime comparisons) and qualitatively (e.g., manual inspection of matched examples), and summarizes findings in the final report.

[1] `https://huggingface.co/datasets/google/wiki40b/tree/refs%2Fconvert%2Fparquet/en`

Students are encouraged to experiment with different parameters (e.g., number of hash functions, number of bands, signature lengths) and analyze their impact on performance. Beyond the required hashing methods, students are also welcome to explore more recent or advanced deduplication techniques, or propose and prototype novel approaches. Innovation and thoughtful system design will be taken into account in the grading rubric, particularly under the criterion of *"comprehensive research and demonstrates innovation."*

4 Tools and References

You may use the following libraries for text preprocessing and vectorization:

- `CountVectorizer`, `TfidfVectorizer`, `HashingVectorizer`, etc. from `sklearn.feature_extraction.text`
- Built-in `hashlib`, `random`, `numpy`, `collections` for hash implementation
- Any other standard Python packages (but not existing LSH libraries)
- Dataset source: **Wiki40B (English)** from Google [2] — you will use the `test` and `validation` subsets.

5 Team Organization

- Students will be assigned into teams of **4 members** each using a random allocation process.
- While team members are expected to divide responsibilities (e.g., feature extraction, hash implementation, evaluation, reporting), each member should maintain an overall understanding of the system.
- All teams are required to submit one final project collectively.

[2] `https://huggingface.co/datasets/google/wiki40b/tree/refs%2Fconvert%2Fparquet/en`

6 Deliverables

Each group is required to submit the following components:

- **Code**: A well-organized Python project with modular structure, inline documentation, and clear instructions for installation and execution (e.g., via `README.md` or Jupyter notebooks).
- **Report**: A concise report (3–5 pages) summarizing the project. The report should include the following sections:
  - **Motivation**: Why deduplication matters and what this project aims to solve.
  - **Related Work**: Brief summary of prior techniques (e.g., MinHash, SimHash, LSH) or relevant systems.
  - **System Design**: Description of the architecture, modules, and key design choices.
  - **Method Comparison**: Analysis of the implemented fingerprinting approaches in terms of performance, time efficiency, and trade-offs.
  - **Experimental Setup and Results**: Dataset setup, parameter settings, analyzed results, and visualizations.
  - **Limitations**: Discussion of observed drawbacks or failure cases.
  - **Conclusion and Future Work**: Summary and potential directions for improvement.
- **Presentation**: Each group will attend a short (8-minute) presentation session with TAs and lecturers to demonstrate their system, findings, and answer questions. Presentation quality will also be considered in grading.

References

[1] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual*

Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 8424–8445, Dublin, Ireland, May 2022. Association for Computational Linguistics.