Algorithm for Massive Data Project: Similar Item Detection in Amazon Book Reviews

Master in Data Science for Economics
Module: Algorithms for Massive Data

Alessandro Cantoni (Mat 13107A)

June 2025

Declaration

1 Introduction

This project addresses the task of detecting similar book reviews within a large dataset by analyzing the textual content of user-generated reviews. Identifying near-duplicate or thematically related reviews can be useful for improving recommendation systems, summarizing feedback, or detecting spam.

The dataset used contains a substantial number of reviews, making scalability a key requirement. To meet this, two approaches are implemented and compared. The first is a brute-force method based on exact Jaccard similarity, which provides full accuracy but suffers from poor scalability due to its quadratic complexity. The second leverages MinHash and Locality Sensitive Hashing (LSH) to provide an efficient, approximate solution that reduces computational cost while preserving most of the relevant matches.

The structure of this report is as follows. Section 2 describes the dataset and the preprocessing techniques applied. Section 3 details the methodologies used for similarity detection. Section 4 presents the experimental results, while Section 5 discusses their implications. Section 6 outlines potential future improvements, and Section 7 concludes the report.

2 Dataset and Preprocessing

The dataset used is the Amazon Books Reviews dataset available on Kaggle under the public domain CC0 license. The file Books_rating.csv is used, which contains user-generated reviews.

A subset of 1000 reviews is extracted. Each review undergoes text preprocessing in order to normalize the content and reduce noise. The following steps are applied:

- **Conversion to lowercase:** to ensure uniformity and eliminate case-based duplicates.
- **Removal of punctuation:** to focus only on meaningful words and discard formatting symbols.
- **Tokenization:** the review text is split into individual words based on whitespace.

- **Stopword removal:** common English words such as “the”, “is”, or “and” are removed using a predefined stopword list.

These steps transform each review into a cleaned and normalized list of tokens, which serves as the basis for further similarity analysis.

3 Methodology

The methodology is based on measuring the similarity between book reviews by comparing their tokenized representations. Two main approaches are implemented: exact pairwise comparison using Jaccard similarity, and a scalable approximation using MinHash and Locality Sensitive Hashing (LSH).

3.1 Jaccard Similarity

Jaccard similarity is a widely used set-based similarity measure defined as the size of the intersection divided by the size of the union of two sets:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

In this project, each review is transformed into a set of tokens (words) after preprocessing. The Jaccard similarity is then computed between all unique pairs of reviews. This brute-force approach is simple and effective but has quadratic time complexity, $O(n^2)$, making it impractical for very large datasets.

3.2 MinHash Signatures

MinHash (short for Min-wise Independent Permutations Hashing) is a probabilistic technique used to efficiently estimate Jaccard similarity between sets. The key idea is to create a compact signature for each set by applying a series of random hash functions and recording the minimum hashed value seen for each.

Let each token in a set be hashed with multiple permutations; for each permutation, the smallest hash value is retained. The resulting signature (a vector of minima) can be used to approximate the Jaccard similarity between sets as the fraction of positions in which the signatures are equal.

In this project, the number of permutations used for MinHash signatures was set to 128. This value represents a balance between estimation accuracy and computational efficiency. Higher values reduce the variance of the Jaccard similarity approximation but increase memory usage and computation time. With 128 permutations, the standard error remains acceptably low, especially for similarity thresholds around 0.3.

The estimated Jaccard similarity using MinHash is defined as:

$$\hat{J}(A, B) = \frac{1}{k} \sum_{i=1}^{k} \delta(h_i(A), h_i(B))$$

where k is the number of hash functions and $h_i(A)$ is the i-th hash of set A. The function $\delta(h_i(A), h_i(B))$ returns 1 if the values are equal and 0 otherwise. This estimation converges to the true Jaccard similarity as k increases.

3.3 Locality Sensitive Hashing (LSH)

Locality Sensitive Hashing is an indexing technique used to group similar items into the same hash buckets with high probability. For MinHash signatures, LSH works by dividing the signature into bands and hashing each band to a bucket.

If two items are similar, they are likely to have the same values in at least one band, and therefore collide in at least one bucket. Only items in the same bucket are compared directly. This significantly reduces the number of comparisons required.

LSH introduces a trade-off between recall and precision depending on the number of bands and rows per band. In this project, LSH is configured using a similarity threshold of 0.3 and uses 128 permutations across all bands.

4 Experiments and Results

4.1 Brute-force Results

- Number of reviews: 1000
- Jaccard threshold: 0.05
- Similar pairs found: 56,665
- Execution time: 55 seconds

4.2 MinHash + LSH Results

- Threshold: 0.3
- Similar pairs found: 582
- Execution time: 11 seconds
- Overlap with brute-force: 520 pairs (89.35%)

4.3 Threshold Sensitivity

Threshold	Similar Pairs Found
0.3	582
0.2	7,321
0.1	16,275

5 Discussion

The brute-force approach provides full recall but at quadratic cost. MinHash + LSH finds most relevant pairs while drastically reducing execution time and comparisons.

Lowering the threshold increases the number of pairs but may reduce precision. LSH effectively handles the trade-off between accuracy and efficiency by drastically reducing the number of pairwise comparisons while preserving most of the truly similar pairs. This is achieved by probabilistically grouping similar items into the same buckets before any similarity calculation is performed. The use of MinHash signatures allows for a fast, approximate estimate of similarity, and LSH ensures that only promising candidates are evaluated in detail. Although LSH may miss some similar pairs (reduced recall), it eliminates most of the unnecessary comparisons (high precision), resulting in a scalable solution for large datasets. The entire pipeline was implemented in Python using open-source libraries and is reproducible via a provided Colab notebook.

6 Limitations and Future Work

The current implementation has been tested on a subset of 1000 reviews. While this is sufficient to validate the correctness and scalability of the solution, larger-scale evaluation could be performed in future work.

Further extensions may include:

- **Using TF-IDF weighting to improve token relevance:** The current approach treats all tokens equally. Term Frequency-Inverse Document Frequency (TF-IDF) could be used to assign more weight to informative words and reduce the impact of frequent, less meaningful ones.
- **Tuning LSH parameters to balance recall and precision:** LSH performance is sensitive to the similarity threshold and the way MinHash signatures are divided into bands. Exploring different settings could improve the trade-off between matching accuracy and computational cost.

7 Conclusion

This project successfully demonstrates a scalable method for detecting similar book reviews based on textual content. Starting from a traditional Jaccard similarity approach, which becomes inefficient at scale, the use of MinHash and Locality Sensitive Hashing (LSH) provides a practical and efficient alternative.

By reducing the number of comparisons and allowing for approximate matching, the system achieves significant gains in performance with minimal loss in accuracy. Experimental results confirm the feasibility of applying this method to larger subsets of the Amazon Books dataset. Although tested on a 1000-review subset, the entire pipeline is designed to scale efficiently to millions of reviews thanks to the use of MinHash and LSH, which drastically reduce both memory usage and computation time.

This methodology is applicable to various domains beyond book reviews, such as detecting near-duplicate documents, spam detection, or clustering customer feedback. Overall, the system balances accuracy and scalability in a way that makes it suitable for real-world deployment in massive text analysis pipelines.