Finding Similar Items

Algorithms for Massive Data (A.Y. 2024/25) – Master in Data Science for Economics

Niccolo' Cibei (46030A)
Università degli Studi di Milano

September 6, 2025

I/We declare that this material, which I/We now submit for assessment, is entirely my/our own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my/our work, and including any code produced using generative AI systems. I/We understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me/us or any other person for assessment on this or any other course of study.

1 Introduction

The goal of this project is to implement a scalable detector of pairs of similar book reviews, using the Amazon Books Review dataset. The dataset contains millions of reviews associated with books sold on Amazon. For each review, several metadata fields are available but this work focuses specifically on the `review/text` field, which contains the raw textual review. Since the dataset is very large, it was implemented a global option to either use the entire collection or to randomly subsample a subset of reviews. This ensures that the code can be executed with limited resources.

The key methodological choice is to encode reviews as sets of tokens and measure similarity using the Jaccard index. To scale this process, we rely on Apache Spark and the MinHash Locality-Sensitive Hashing (LSH) technique, which allows approximate similarity joins over large collections of documents. This approach ensures that the solution is both computationally efficient and scalable to massive datasets.

2 Preprocessing Techniques

Before building the similarity detector, the textual reviews must be cleaned and transformed into a suitable representation. The preprocessing pipeline consists of the following steps, implemented in Apache Spark:

- **Data loading and filtering.** The `Books_rating.csv` file was loaded using Spark's CSV reader. Only the `Id` and `review/text` columns are retained. Reviews with missing or empty text are discarded.

- **Tokenization.** Each review is split into tokens using a regular expression–based tokenizer. The tokenizer separates words by non-alphanumeric characters and converts them to lowercase. This ensures that words such as "Book," "book," and "BOOK" are treated uniformly.
- **Stopword removal.** Common English stopwords (e.g., *the*,*and*, *of*) are removed using Spark's built-in `StopWordsRemover`. Eliminating these high-frequency but semantically uninformative tokens reduces noise and improves the robustness of the Jaccard similarity measure.
- **Filtering short reviews.** Reviews containing fewer than four distinct tokens are excluded. Very short texts (e.g., "Great book!") do not provide enough information for reliable similarity analysis and would otherwise inflate the number of trivial matches.
- **Feature representation.** The cleaned tokens are mapped into high-dimensional binary vectors using the `HashingTF` transformer. Each dimension corresponds to a hash bucket, and the presence of a token is encoded as 1. This representation enables efficient MinHashing and approximate similarity joins at scale.

This preprocessing pipeline ensures that the reviews are transformed from raw text into compact, standardized feature vectors suitable for large-scale similarity analysis.

3 Implementation

3.1 MinHashLSH for Candidate Generation

After preprocessing, each review is represented as a high-dimensional binary vector indicating the presence of tokens. To avoid the quadratic complexity of computing pairwise Jaccard similarity across all reviews, we employ `MinHashLSH`, an approximate nearest-neighbor method available in Spark ML. This technique projects the hashed vectors into compact signatures and enables efficient similarity joins. Given a distance threshold, MinHashLSH quickly retrieves candidate pairs of reviews that are likely to be similar.

3.2 Exact Jaccard Similarity

For each candidate pair produced by MinHashLSH, we compute the exact Jaccard similarity based on the token sets:

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

where A and B are the sets of distinct tokens in the two reviews. Only pairs whose similarity exceeds a configurable threshold (default 0.2) are retained. The system can also extract the top-k most similar pairs, which are then used for evaluation and inspection.

3.3 Scalability and Limitations

The implementation is designed to scale to massive datasets. Using Apache Spark, both preprocessing and similarity search are distributed across multiple nodes. The hashing

trick ensures constant-size feature vectors regardless of vocabulary, while MinHashLSH drastically reduces the number of pairwise comparisons. This makes the approach feasible even for millions of reviews.

However, several limitations must be noted. First, LSH is an approximate method: some highly similar pairs may be missed (false negatives), while some returned pairs may be spurious (false positives). Second, the method is purely token-based and does not account for semantic similarity (e.g., "great" and "excellent" are considered unrelated). Finally, large thresholds or very high-dimensional hash spaces can still lead to memory and runtime overheads, despite Spark's distributed architecture.

Overall, the combination of token-based Jaccard similarity with MinHashLSH offers an efficient and scalable baseline solution for detecting near-duplicate reviews. yep

4 Experiment Results

4.1 Dataset Statistics

We conducted experiments on a subsample of the dataset of 500 reviews. The system successfully identified 5 pairs of reviews with lexical overlap. We used a Jaccard threshold of 0.2(see Table 1).

4.2 Effect of Similarity Threshold

There is the possibility to vary the Jaccard similarity threshold to observe its effect on the number and quality of detected pairs. At low thresholds, many pairs were retrieved, including weakly related reviews. Increasing the threshold to 0.3 or 0.4 produced fewer pairs, but with stronger lexical overlap and higher quality. This trade-off highlights the role of the threshold in balancing recall and precision.

4.3 Runtime and Scalability

To test scalability, we measured runtime on increasing dataset sizes. Thanks to Spark's distributed execution and the use of MinHashLSH, the runtime scaled sublinearly compared to the quadratic cost of naive pairwise comparison. For example, analyzing 500 reviews required approximately 1 minute and a half.

4.4 Summary of Findings

Overall, the experiments confirm that the proposed pipeline is effective at detecting near-duplicate reviews at scale. The approach is efficient, produces interpretable results, and demonstrates the trade-off between threshold setting and quality of matches.

5 Conclusions

In this project we developed a scalable system for detecting pairs of similar book reviews from the Amazon Books dataset. The solution is based on token-level Jaccard similarity combined with MinHash locality-sensitive hashing (LSH), implemented in Apache Spark. This design enables efficient candidate generation and filtering.

Table 1: Top similar review pairs at Jaccard threshold 0.2 (subsample run).

Jaccard	**Text A**	**Text B**
0.231	The book arrived very early in promised condition. Would be pleased to do business with this seller again. Thanks	the book arrived within 3 weeks after my purchase. book was in a good condition too.
0.231	Well written, this book is both scholarly and informative in it's insight and portrayal of Maximilian Kolbe, a Saint for our time.	Well-written book on an unusual topic
0.222	This is a good introductory book. Comprehensive and well organized	Well-written book on an unusual topic
0.214	This book was very informative and well written. It should be read by every one who wants to know the truth about tithing and giving.	Well-written book on an unusual topic
0.211	Well written, this book is both scholarly and informative in it's insight and portrayal of Maximilian Kolbe, a Saint for our time.	This book was very informative and well written. It should be read by every one who wants to know the truth about tithing and giving.

The experiments showed that the method effectively identifies duplicate and near-duplicate reviews, with the similarity threshold providing a natural way to trade off recall and precision. Lower thresholds produce more candidate pairs, but also include weaker similarities, while higher thresholds result in fewer but stronger matches.

Despite its strengths, the approach has some limitations. Since the similarity measure is purely token-based, it does not capture semantic relations between words (e.g., "great" vs. "excellent"). Moreover, MinHashLSH is approximate, meaning that some similar pairs may be missed, while some spurious pairs may still be retrieved.

Overall, the project demonstrates how classical similarity measures combined with scalable algorithms can effectively solve large-scale text similarity problems.