Project: FINDING SIMILAR ITEMS

Ulan Shaikyp (944462)

Master in Data Science for Economics

December 2025

Università degli Studi di Milano

Declaration of Originality

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, except to the extent that such work has been cited and acknowledged within the text, including any code produced using generative AI systems. I/We understand that plagiarism, collusion, and copying are serious academic offences and accept the penalties that may be imposed should I/We engage in such practices. 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.

Contents

List of Figures

1 Introduction

In this project, near-duplicate text detection is addressed in the context of large-scale user-generated data. Near-duplicate reviews may result from spam, repeated submissions, or minor textual variations and can negatively affect the quality of analysis. A direct pairwise comparison of all reviews is computationally infeasible for large datasets due to its quadratic time complexity. To overcome this limitation, MinHash and Locality-Sensitive Hashing (LSH) are applied to efficiently identify near-duplicate reviews in the Amazon Books Reviews dataset by approximating Jaccard similarity. This approach reduces the computational complexity to approximately $O(n)$ in practice while maintaining high precision. The implementation follows the methodology described in Chapter 3 of *Mining of Massive Datasets*.

2 Dataset Description

2.1 Source and Sampling

This study uses the *Amazon Books Reviews* dataset obtained from *Kaggle*. The complete dataset contains approximately 3 million customer reviews. Due to its large size, performing exhaustive pairwise comparisons across all reviews is not feasible. In particular, comparing every review with every other review would require more than 1.2 billion comparisons, which would take an impractical amount of time on standard hardware.

To make the analysis computationally manageable while maintaining reliable results, different sample sizes were tested. Based on these tests, a random sample of 50,000 reviews was selected as the best compromise between efficiency and analytical quality. Larger samples significantly increased runtime and memory usage, while smaller samples provided less meaningful results.

The sample was drawn randomly using a fixed random seed (*seed = 42*). This ensures that the same subset of reviews can be reused in repeated experiments, improving reproducibility.

2.2 Dataset Attributes

Each review in the dataset includes textual content and metadata. The attributes used in this study are:

- **review/text**: the full text of the review;
- **review/summary**: a short summary or title of the review;
- **title**: the title of the reviewed book;
- **user id**: an anonymized identifier of the reviewer;
- **review/score**: a numerical rating from 1 to 5 stars;

Table 1: Distribution of Review Ratings

Rating	Count
1.0	3,369
2.0	2,557
3.0	4,261
4.0	9,677
5.0	30,136

- **review/helpfulness**: a score indicating how helpful the review was to other users.

2.3 Sample Characteristics

The sampled dataset consists of 50,000 reviews written by 35,350 unique users and covering 24,285 different books. Before preprocessing, the average review length is 143.38 words, which provides sufficient textual information for similarity analysis.

Table 1 shows the distribution of review ratings. Most reviews are positive, with 4- and 5-star ratings accounting for the majority of the data. This pattern is common in online review platforms and is considered in the analysis.

After describing the dataset, the next step is data cleaning and text preprocessing.

3 Preprocessing

This chapter describes the preprocessing steps applied to the dataset in order to standardize the textual content, reduce noise, and prepare the data for similarity detection.

3.1 Text Cleaning

All reviews were subjected to a sequence of text normalization procedures. First, HTML entities (e.g., `&`, `&quot;`) were decoded into their Unicode representations using Python's `html.unescape()` function. Second, all text was converted to lowercase to ensure case-insensitive matching. Third, non-alphanumeric characters, with the exception of spaces, were removed using regular expressions. Finally, consecutive whitespace characters were collapsed into a single space, and leading and trailing whitespace was removed.

3.2 Stop Word Removal

To reduce dimensionality while preserving semantic information, a predefined list of 30 common English stop words was removed from the text. This list

includes articles, conjunctions, prepositions, common verbs, and personal pronouns (e.g., *the, and, in, is, this*).

3.3 Filtering and Deduplication

Additional filtering was performed to ensure the suitability of the data for shingling-based similarity detection. Reviews containing fewer than five words after stop word removal were discarded in order to guarantee the generation of at least three 3-word shingles ($k = 3$). Furthermore, exact duplicates—defined as reviews with identical cleaned text, book title, and user identifier—were removed, resulting in the exclusion of 215 reviews. Reviews with identical text but differing metadata were retained, as they represent valid cases for near-duplicate detection; 1,301 such reviews were preserved.

After preprocessing, the final dataset comprised **49,744 reviews**.

4 Methodology

The near-duplicate detection pipeline consists of four sequential stages: shingling, MinHash signature generation, Locality-Sensitive Hashing (LSH), and candidate verification using Jaccard similarity.

4.1 Shingling

Each review was represented as a set of contiguous word shingles with shingle size $k = 3$. This value was chosen to balance specificity and generalization: larger values of k increase precision but may miss semantically similar reviews, while smaller values increase the number of false positives. After preprocessing, reviews were converted into their corresponding shingle sets. In the sampled dataset, the number of shingles per review ranged from 2 to 3,597.

4.2 MinHash Signatures

MinHash signatures were generated using $m = 128$ hash functions, providing a good trade-off between similarity approximation accuracy and computational efficiency. The `datasketch` Python library was used, which implements MinHash via random hash functions. This step reduced each review from a variable-size shingle set (average size 94.5 shingles) to a fixed-size 128-dimensional signature, enabling efficient storage and comparison.

4.3 Locality-Sensitive Hashing

An LSH index was constructed using the 128-dimensional MinHash signatures with a similarity threshold of $t = 0.7$. The `datasketch.MinHashLSH` implementation automatically configures banding parameters to optimize candidate selection for the specified threshold. All 49,744 signatures were inserted into the index, resulting in 1,530 candidate pairs retrieved for further verification.

4.4 Jaccard Similarity

The Jaccard similarity coefficient was used to measure the true similarity between two reviews and is defined as:

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

where A and B denote the shingle sets of two reviews. The coefficient ranges from 0 (disjoint sets) to 1 (identical sets).

4.5 Candidate Verification

Since MinHash provides only an approximation of Jaccard similarity, all candidate pairs identified by LSH were verified by computing the exact Jaccard similarity from the original shingle sets. For each candidate pair, the intersection and union of shingles were computed, and pairs with similarity greater than or equal to 0.7 were retained.

Out of the 1,530 candidate pairs generated by LSH, 764 pairs (50.0%) satisfied the verification criterion. The remaining candidates were false positives. This verification rate indicates that the LSH configuration effectively balanced candidate reduction and precision.

5 Experimental Results

5.1 Algorithm Performance

The proposed pipeline achieved substantial reduction in comparison complexity. While brute-force comparison would require 1,193,207,896 comparisons, the LSH stage generated only 1,530 candidate pairs. The resulting bucket structure was highly sparse: only 1,330 reviews (2.67%) had at least one candidate, with an average of 0.03 candidates per review and a maximum of 8.

Exact Jaccard similarity verification retained 764 pairs (49.9%), while 766 candidates were discarded as false positives. This verification rate indicates an effective LSH configuration, balancing candidate reduction and accuracy.

5.2 Similarity Distribution

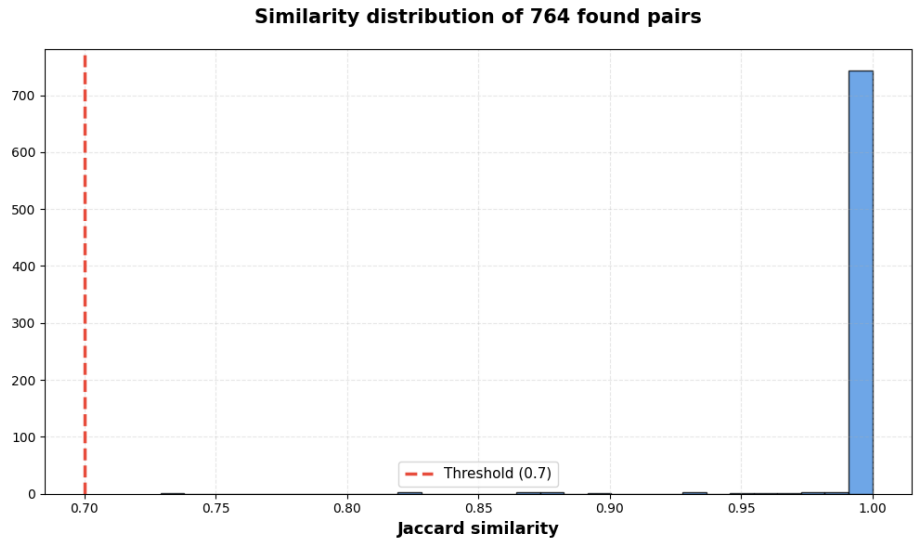

Figure 1: Similarity Distribution histogram

The distribution of Jaccard similarity scores for the detected pairs was strongly right-skewed. A total of 755 pairs (98.8%) exhibited similarity scores above 0.9, with 740 pairs showing perfect similarity. Moderate similarity values (0.7–0.8) were rare, supporting the choice of a conservative similarity threshold.

Category breakdown: • Near-Duplicates (¿0.9): 755 pairs • High Similarity (0.8–0.9): 8 pairs • Substantial Similarity (0.7–0.8): 1 pair

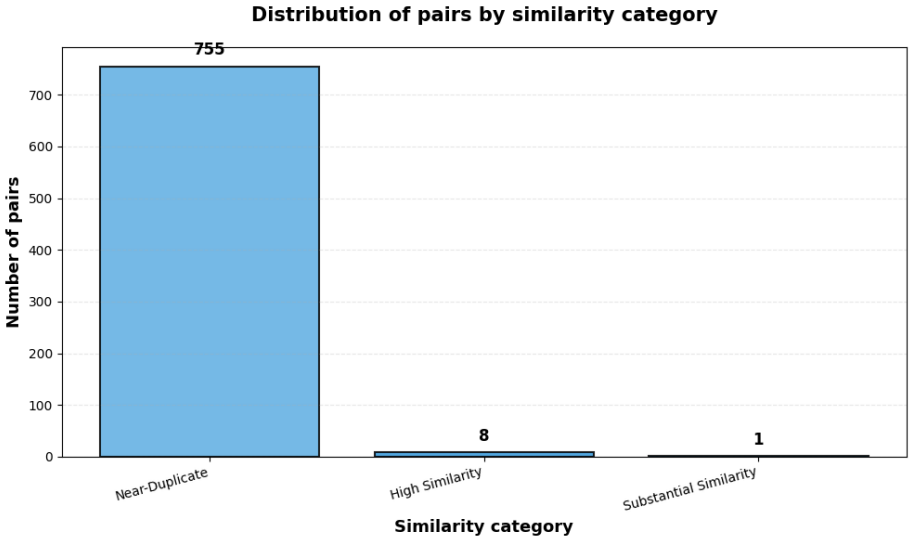

Figure 2: Category Distribution bar chart

5.3 User and Book Analysis

Most similar pairs involved the same user posting identical reviews for different books (643 pairs), typically corresponding to different editions or books within a series. Cross-user near-duplicates were uncommon, suggesting limited plagiarism and predominantly legitimate content reuse. Analysis of book similarity revealed strong clustering among different editions of classic works, reflecting consistent user behavior across editions.

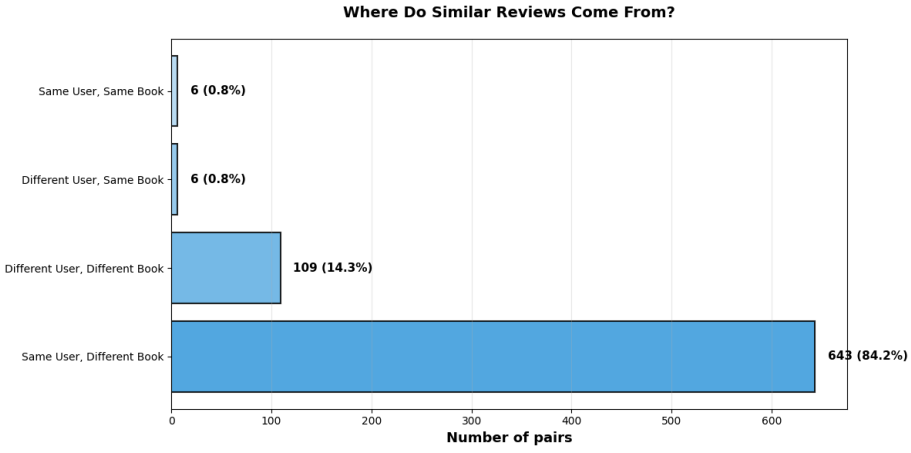

Figure 3: User/Book Relationship horizontal bar chart

5.4 Book similarity network

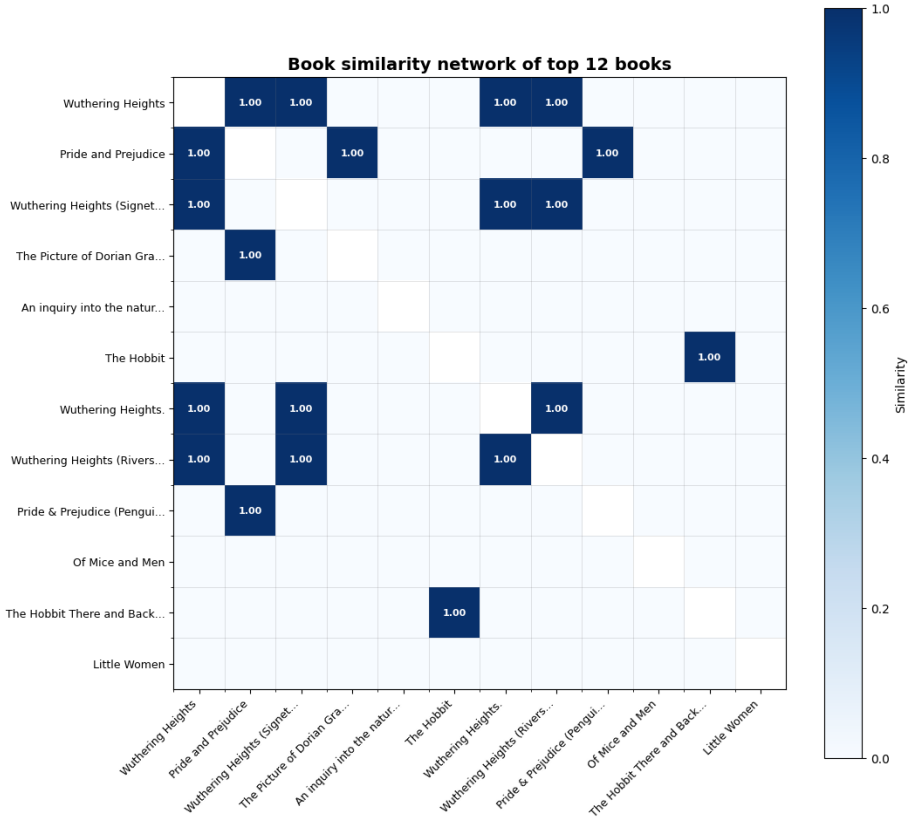

Figure 4: Book Similarity Matrix heatmap

Here, a similarity matrix of the twelve books most frequently appearing in near-duplicate pairs is shown. Strong clustering is observed among different editions of classic works (e.g., *Wuthering Heights*, *Pride and Prejudice*, *The Hobbit*), with all corresponding pairs exhibiting perfect similarity (1.0000). This pattern indicates systematic reuse of identical review text across multiple editions of the same book. In contrast, some books (e.g., *Little Women*, *Of Mice and Men*) appear frequently but show no connections to other top-ranked books, as their similar pairs involve less frequent titles outside the top-twelve set.

5.5 Quality Assessment

Manual inspection of the top-ranked pairs confirmed that all examined matches represented genuine near-duplicates, with similarity scores above 0.8, validating the correctness of the approach.

6 Conclusion

This project demonstrates that MinHash combined with Locality-Sensitive Hashing provides an efficient and accurate solution for near-duplicate detection in large-scale review datasets. Using 49,744 Amazon book reviews, the method identified 764 near-duplicate pairs while examining only 1,530 candidates. The results reveal that near-duplicates primarily arise from users reusing their own reviews across different books or editions, rather than from widespread cross-user plagiarism. Overall, the approach achieved high precision, strong computational efficiency, and meaningful insights into user reviewing behavior, confirming its suitability for large-scale text similarity analysis.