Finding Similar Items using Jaccard Similarity

Aknur Abdikarimova, 33666A
Data Science for Economics, University of Milan
aknur.abdikarimova@studenti.unimi.it

I declare that this material, which I now submit for assessment, is entirely my 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 work, and including any code produced using generative AI systems. I 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 or any other person for assessment on this or any other course of study.

1. Introduction

Online reviews play a huge role in how people choose what to buy. On platforms like Amazon, thousands of users leave book reviews every day — but not all of them are unique. Many reviews are repeated, copied from others, or slightly rewritten versions of the same text. These duplicate and near-duplicate reviews can easily distort product ratings and create misleading impressions about a book's popularity or quality.

The goal of this project is to detect similar book reviews in the Amazon Books Reviews dataset from Kaggle. The dataset includes the book title, user rating, and two main text fields: a short summary and the full review text. To make analysis easier, both fields are combined into one text column and cleaned from unnecessary symbols, HTML tags, and other noise. Only English-language reviews are kept for further processing.

To find similar texts efficiently, I use a combination of shingling, MinHash, and Locality-Sensitive Hashing (LSH). In simple terms:

- **Shingling** breaks each review into small word sequences (like sliding windows).
- **MinHash** creates compact signatures of those word sets.
- **LSH** makes it possible to quickly find reviews that have similar signatures — without comparing every single pair directly.

This method makes it possible to analyze tens of thousands of reviews without running into performance issues. The detected pairs are then grouped into categories such as duplicates, near-duplicates, high-similarity, and related.

Overall, the project shows how simple text-processing techniques combined with efficient hashing can help clean large review datasets, reduce redundancy, and highlight interesting patterns in how people write about books.

2. Dataset Description

The dataset used in this project is the Amazon Books Reviews dataset published on Kaggle by Mohamed Bakhet. It contains a large number of user reviews for books sold on Amazon, covering a wide range of titles and review types. Each entry represents one review with both textual and numerical information.

Main characteristics of the dataset:

- Total number of rows (book id): 3,000,000

- Total number of columns: 10
- Unique book titles: 212,404
- Unique review summaries: 1,592,316
- Unique full review texts: 2,062,649

Key columns used in this project:

- `id` — the identifier of book (not unique)
- `title` — the title of the book being reviewed
- `review/score` — the numerical star rating (from 1 to 5)
- `review/summary` — a short description or headline of the review
- `review/text` — the main body of the review text

The dataset does not provide a standardized book identifier (such as ASIN). It appears to be a scraped or arbitrary value occasionally reused across multiple titles. Therefore, the title field is used to group reviews by book, which can occasionally lead to overlap when different editions share the same title.

For this analysis, only the text-related columns and key metadata were used. The `review/summary` and `review/text` columns were merged into a single text column to create a unified representation of each review before preprocessing and similarity detection.

https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews/data

3. Data Preprocessing and Normalization

Before comparing reviews, the text needed to be cleaned, standardized, and transformed into a structured format suitable for similarity analysis. The full preprocessing pipeline included the following steps:

- **Merge text fields:** combined `review/summary` and `review/text` into a single column `text` to capture both short and detailed review content.
- **Clean text:** removed HTML tags, special characters, and punctuation using regular expressions. All text was converted to lowercase and extra spaces were stripped out.

- **Language filtering:** used `langdetect` library to keep only English reviews longer than a minimum length of 20 words. This helped remove noise and non-English entries.
- **Word filtering:** excluded reviews with fewer than five words to focus on meaningful texts.
- **Random sampling:** selected around 30,000 reviews from the first 100,000 to balance variety and processing time.
- **Tokenization and lemmatization:** split each review into individual words (tokens), converted words to their base form using `WordNetLemmatizer` library, and removed English stopwords such as *the*, *is*, *and*, etc.
- **Shingling:** generated small overlapping word sequences (3-word shingles) from each processed review. These shingles form the basis for calculating textual similarity.
- **Common-term filtering:** removed shingles that appear in more than 10% of all documents to avoid generic, uninformative patterns.

After these steps, each review was transformed into a normalized list of informative shingles. This representation allows efficient comparison of texts based on word overlap while minimizing noise from formatting, grammar, or stopwords.

4. Algorithms

To identify similar or duplicate reviews efficiently within a large dataset, the project applies a combination of MinHash and Locality-Sensitive Hashing (LSH). These algorithms approximate the Jaccard similarity between documents while avoiding the high computational cost of comparing every possible pair directly.

4.1 MinHash

Each review is represented as a set of 3-word shingles, which are small, consecutive sequences of words. The Jaccard similarity between two reviews measures how much these sets overlap, but computing it for millions of reviews would be prohibitively expensive.

MinHash provides a compact numerical representation of each document's shingle set. It applies multiple independent hash functions to the shingles and keeps only the minimum hash value from each function. The resulting vector of minimum values — called a

MinHash signature — acts as a short fingerprint of the document. Reviews with similar shingle sets produce similar signatures.

4.2 Locality-Sensitive Hashing (LSH)

Once MinHash signatures are computed for all reviews, LSH is used to efficiently retrieve candidate pairs that are likely to be similar. LSH divides each signature into several bands and hashes each band into a bucket. Reviews that fall into the same bucket in at least one band are treated as candidate pairs for further checking.

This approach drastically reduces computation time — from quadratic complexity $O(n^2)$ to approximately linear $O(n)$ — by focusing only on pairs that have a high probability of being similar.

4.3 Implementation Details

In this project:

- Each review's shingles were hashed using the `mmh3` hash function.
- 64 hash permutations were used to generate the MinHash signatures.
- The similarity threshold for the LSH stage was set to 0.6, meaning that only reviews with an estimated Jaccard similarity of 60% or higher were considered as potential matches.
- All retrieved candidate pairs were then verified using the exact Jaccard similarity to confirm whether they truly met the threshold.

The combined MinHash–LSH method allows efficient and scalable detection of near-duplicate and highly similar reviews, even in datasets containing millions of documents.

5. Results and Analysis

After applying the MinHash–LSH pipeline to the preprocessed Amazon Books Reviews dataset, a total of 831 candidate pairs were identified as similar based on their Jaccard similarity scores. Each pair corresponds to two reviews that share a significant overlap in tokenized content.

5.1 Candidate Pair Distribution

All detected candidate pairs were categorized into qualitative similarity levels based on their Jaccard similarity scores. Each category corresponds to a specific similarity range, reflecting how closely two reviews resemble each other.

Table 1: Distribution of Candidate Pairs by Similarity Category

Category	**Jaccard Range**	**Count**
Duplicate	1.0	736
Near-duplicate	0.9 – 1.0	55
High-similarity	0.8 – 0.9	21
Related	0.7 – 0.8	11
Loose	≥ 0.6	8
Total candidate pairs	—	**831**

The results show that most detected pairs fall into the duplicate and near-duplicate categories, confirming that many reviews in the dataset are either exact repetitions or contain minimal differences such as punctuation, spacing, or casing. The smaller number of pairs in the high-similarity and related groups reflects genuine textual variations and paraphrasing across different reviewers, while loose pairs represent weak overlaps near the similarity threshold.

5.2 Jaccard Similarity Distribution

The overall distribution of Jaccard values is shown in Figure 1.

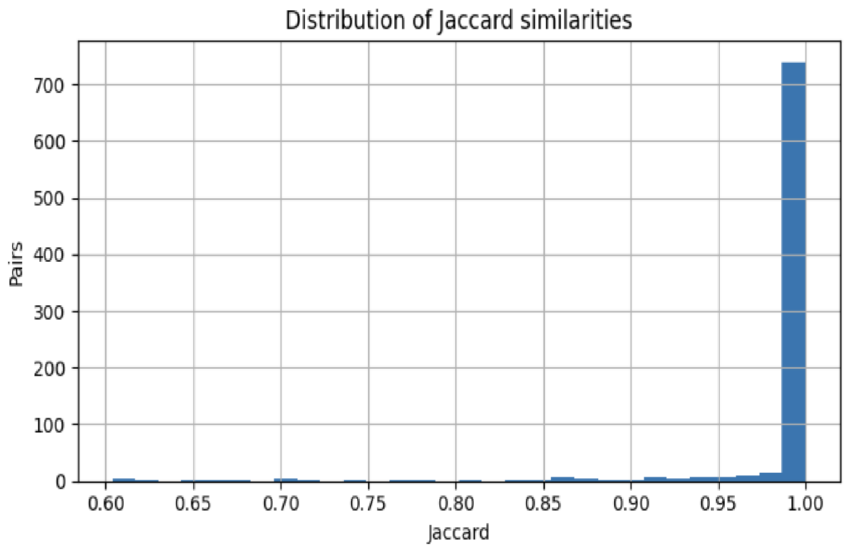

Figure 1: Distribution of Jaccard Similarities

The histogram reveals a strong concentration of values near 1.0, confirming that most pairs exhibit very high lexical overlap. Only a small fraction of pairs fall below 0.9, which indicates that the LSH algorithm effectively identifies tight clusters of highly similar reviews while filtering out unrelated text pairs.

5.3 Representative Candidate Examples

To better illustrate how the categories differ, Table 2 presents one representative example per similarity level. Excerpts are shortened for readability.

Table 2: Representative Candidate Pairs by Similarity Category

Category	**Jaccard score**	**Title 1**	**Title 2**	**Review 1**	**Review 2**
Duplicate	1.00	Fahrenheit 451	Fahrenheit 451	"amazingly relevant for our times... an excellently crafted piece that will go down in history as a classic."	"amazingly relevant for our times... an excellently crafted piece that will go down in history as a classic."
Near-Duplicate	0.9896	Roman Catholicism	Roman Catholicism	"a hate book written by an author with made up facts and unsubstantiated claims first off let me say that i have no ax to grind in reviewing ... "	"this is nothing more than a hate book written by an author who has made up facts together with unsubstantiated claims ... "
High Similarity	0.8947	Foundation	The Martian Way	"wonderful book from a wonderful author asimov's foundation books are well crafted and populated with fascinating characters... "	"fantastic concept well executed asimov's foundation books are well crafted and imagined with fascinating characters... "
Related	0.7778	Great Expectations	The Scarlet Letter: A Romance	"classic it was a freebie paperless book why not take advantage of it... "	"more freebie it was a paperless book why not take advantage of it... "
Loose	0.6289	A Princess of Mars	A Princess of Mars	"carter jump to mars and adventures start kindle edition of erb s martian series edgar rice burroughs 1875 1950 was a prodigy of imagination... "	"first brick in the martian wall edgar rice burroughs 1875 1950 was a prodigy of imagination he started his writer career quite late his first work was published in 1912... "

Interpretation and Observations:

- **Duplicate and Near-Duplicate Reviews:** These constitute the majority of detected pairs. In many cases, reviews were identical except for spacing or punctuation. This suggests either reposting across editions or data duplication during scraping.
- **High-Similarity and Related Reviews:** Pairs in these categories share thematic overlap but not identical wording — for example, reviews describing the same author or series with slightly different phrasing. These are useful for identifying rephrased or paraphrased content.

- **Loose Similarity:** These pairs occur close to the similarity threshold, often reflecting partial lexical overlap rather than clear duplication. They demonstrate the sensitivity of the Jaccard metric to token-level variation.

5.4 Review Score Patterns

To explore whether duplicated or highly similar reviews exhibit systematic differences in ratings, Figure 2 presents the distribution of review scores across similarity categories.

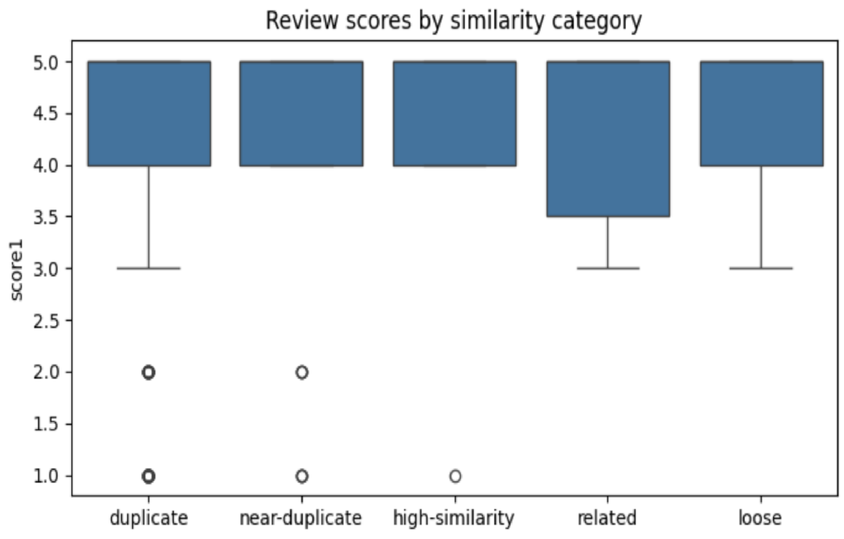

Figure 2: Review Scores by Similarity Category

Across all similarity levels, review scores remain strongly concentrated between 4 and 5 stars. Duplicates and near-duplicates display minimal variation, suggesting that repeated reviews often preserve not only wording but also sentiment. Lower-rated outliers (1–2 stars) appear occasionally, mostly among duplicates, likely reflecting exact copies of negative reviews rather than genuine diversity of opinion.

5.5 Books with the Most Duplicate or Near-Duplicate Reviews

The titles most affected by repetitive reviews are shown in Figure 3.

The results highlight that duplication is not uniformly distributed across the dataset — instead, it is concentrated around a few popular titles.

The books *Fahrenheit 451* and *Foundation* show the highest duplication levels, with more than 100 paired duplicates each. These are well-known classics that often appear in multiple editions and reprints, which likely contributes to repeated reviews being copied across versions. Similarly, *The Martian Way* and *The Mayor of Casterbridge* also exhibit

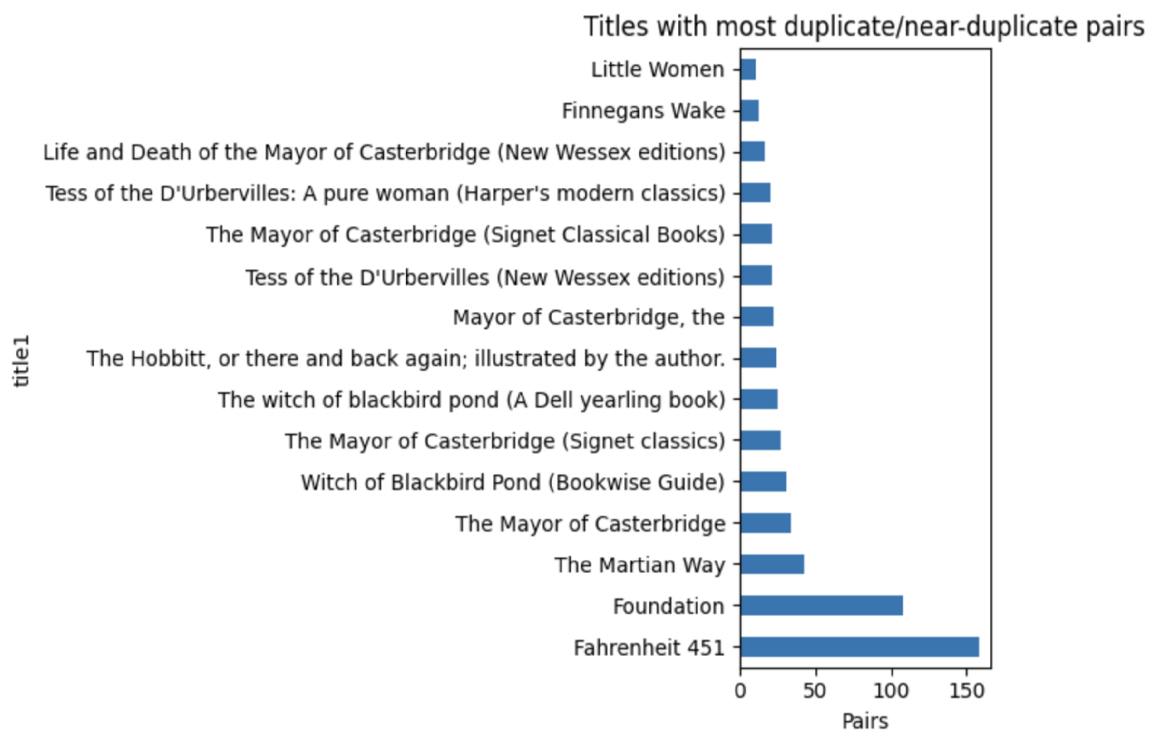

Figure 3: Titles with Most Duplicate/Near-Duplicate Pairs

substantial redundancy, suggesting the same review text was reused or slightly modified across editions or by different users.

The long tail of titles (e.g., *Little Women*, *Finnegans Wake*, *The Hobbit*) with smaller but still notable duplication counts reinforces a pattern where review duplication correlates with book popularity and multiple-edition presence.

Overall, this visualization demonstrates that duplicate content in the dataset is not random but rather clustered around frequently reviewed literary works, which has important implications for data cleaning and bias reduction in sentiment or recommendation models trained on this data.

5.6 Analysis of Cross-Title Duplicates

The cross-title duplicates likely occur because different books by the same author (for example, Isaac Asimov's *Foundation* and *The Martian Way*) often share similar reader bases and are listed together in collections or series. As a result, users may reuse identical reviews across related titles or copy text automatically when books are republished under slightly different editions. This pattern reflects how author-level or edition-level overlap can introduce non-unique reviews into the dataset, underscoring the need for duplicate detection during preprocessing.

6. Improvement

A practical next step would be to automatically remove exact duplicate rows — cases where all column values given in the original dataset are identical — to reduce redundancy before similarity analysis. Additionally, the project could be improved by standardizing book titles such as removing edition notes or punctuation to better group reviews referring to the same work.

7. Conclusion

This project explored the problem of detecting duplicate and near-duplicate book reviews in the Amazon Books dataset using MinHash and Locality Sensitive Hashing (LSH). Through systematic preprocessing, text normalization, and approximate similarity search, it was possible to efficiently identify hundreds of pairs of highly similar reviews within a sample of the data. The results confirmed that duplication is a widespread phenomenon — not only across reviews of the same title but also between books by the same author, likely due to series editions, reused metadata, or reader behavior.

The analysis also revealed that many identical reviews share the same sentiment and score, reinforcing the idea that these duplicates are genuine repetitions rather than coincidental similarities. This insight is especially relevant for large-scale text analytics, where redundancy can bias sentiment models and inflate similarity-based recommendations.

The MinHash–LSH framework proved to be both scalable and effective, capable of processing large datasets while maintaining reasonable computational efficiency. However, the project also highlighted the need for continued improvements — such as removing exact duplicates, standardizing titles, and incorporating semantic similarity methods to capture paraphrased or contextually similar reviews.

Overall, this work demonstrates that efficient text deduplication is an essential preprocessing step for ensuring data quality and reliability in machine learning applications. It provides a foundation that can be expanded toward more sophisticated, hybrid similarity detection systems combining lexical and semantic approaches.