Finding Similar Items in Amazon Book Reviews

Selen Gezginci
32204A
Master in Data Science for Economics

Academic Year 2024/2025

Declaration

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

The aim of this project is to identify pairs of similar book reviews in a large Amazon dataset by using Jaccard similarity to process the text column. In this project, advanced algorithms like MinHashing and Locality Sensitive Hashing (LSH) were used to encode reviews and find candidate pairs to approximate Jaccard similarity.

2 Dataset Information

The dataset used is the Amazon Books Reviews dataset obtained from Kaggle. Each row includes user reviews with a 'review/text' field containing free-text book feedback. The dataset contains 3 million rows.

Column Name	**Data Type**	**Description**
Id	String	The unique Amazon Standard Identification Number (ASIN) for the book. Note that multiple reviews can share the same Id if they refer to the same book.
Title	String	The full title of the book being reviewed.
Price	String	The price of the book at the time of data collection (may contain null values).
User_id	String	A unique identifier for the user who wrote the review.
profileName	String	The public display name of the reviewer.
review/helpfulness	String	A fraction (e.g., "2/3") indicating how many users found the review helpful out of the total number of voters.
review/score	Double / String	The rating assigned to the book by the reviewer, on a scale from 1.0 to 5.0.
review/time	Long / String	The Unix timestamp indicating when the review was posted.
review/summary	String	A short, title-like summary of the review content (e.g., "Great read!").
review/text	String	The full body text of the review. This is the primary feature used for similarity detection in this project.

3 Exploratory Data Analysis (EDA)

Before implementing the similarity detection algorithms, a comprehensive exploratory analysis was shown using PySpark SQL and visualization libraries (Matplotlib/Seaborn) to understand the dataset's structure, distribution, and quality.

Top 5 Most Reviewed Books

Title	review_count	avg_rating
The Hobbit	22023	4.66
Pride and Prejudice	20371	4.53
Atlas Shrugged	12513	4.03
Wuthering Heights	10780	4.05
The Giver	7644	4.27

Figure 1: Top 5 most reviewed books in the dataset, along with their total number of reviews and average user ratings.

This figure highlights the most frequently reviewed books, indicating that highly popular books tend to receive a large volume of user feedback while maintaining relatively high average ratings.

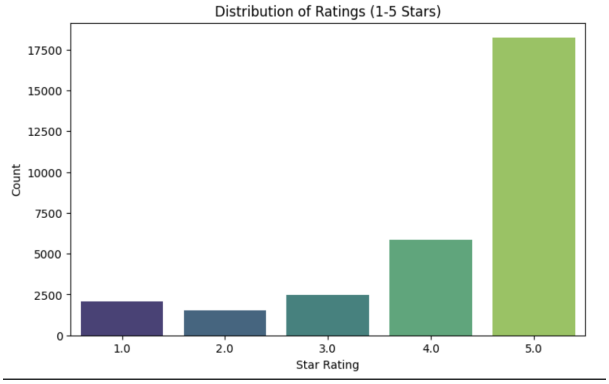

Figure 2: Distribution of Ratings

Rating Distribution

rating	count	percentage
5.0	1795786	60.22
4.0	581728	19.51
3.0	252940	8.48
2.0	150449	5.05
1.0	201000	6.74

Figure 3: Distribution of user ratings across the dataset.

Table represents the distribution of user ratings across the dataset. The results show a strong bias toward positive reviews , with 5-star ratings accounting for 60.22% of all reviews (nearly 1.8 million). When combined with 4-star ratings (19.51%), positive reviews represent approximately 80% of the dataset. On the other hand, negative ratings (1 and 2 stars) form a relatively small minority, comprising less than 12% of all reviews.

Review Length Analysis

length_bucket	avg_rating	count
Medium (100–1000)	4.23	2346289
Long (>1000)	4.14	501980
Short (20–100)	4.29	128679
Very Short (<20 chars)	4.3	4955

Figure 4: Analysis of review length categories, presenting the average rating and number of reviews for different review length ranges.

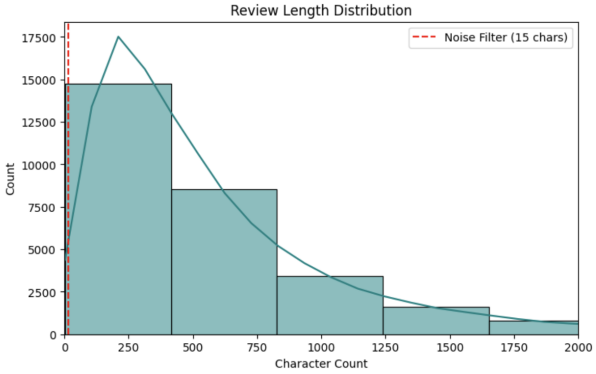

Figure 5: Review Length Distribution

Reviews are classified according to their textual length, from very short comments to long-form reviews. The distribution shows that the dataset is dominated by short and medium-length reviews.

4 Data Preprocessing

Befor to similarity computations, the book review texts were transformed through a preprocessing process designed to reduce noise and standardize the input for simlarity measures.

4.1 Data Sampling

Given the large volume of the original dataset (3 million records), processing the entire corpus in memory was computationally prohibitive. A random sample was taken for analysis. Sample Fraction: (10%) of the total dataset.

4.2 Text Cleaning Pipeline

4.2.1 HTML Decoding

Reviews scraped from the web often contain encoded entities such as &amp, &quot, or &gt. These are decoded into their corresponding characters to prevent the algorithm from treating formatting artifacts as meaningful content.

4.2.2 Case Normalization

All characters are converted to lowercase. This ensures that "Great Read" and "great read" are seen as same, so the algorithm looks at the meaning of the words.

4.2.3 Digit Normalization

All numeric digits are changed to a generic zero (0). For deduplication purposes, the difference between "Page 105" and "Page 106" is often irrelevant (e.g., they might be different editions of the same book). Generalizing them to "Page 000" allows the system to detect structural duplicates.

4.2.4 Punctuation Removal

Removed all non-alphanumeric characters (punctuation and special symbols).

4.2.5 Whitespace Elimination

Stripped all spaces, tabs, and newlines. This ensures that the shingling process focuses purely on the character sequence, making it robust against formatting differences (e.g., double spaces).

4.2.6 Length Filtering

Reviews resulting in fewer than 30 characters after cleaning were discarded.

4.3 Shingling (k-Shingling)

Shingling is a technique used to convert a document into a set of fixed-length substrings, formally known as k-shingles or *character- n-grams.*This representation transforms a continuous text into a mathematical set, allowing for set-based similarity comparisons (like Jaccard Similarity). In this project, I applied **character-level shingling with** $k = 6$ Before to shingling, the text is normalized by converting to lowercase and removing all whitespace and punctuation. For example, consider the word "fantastic". After preprocessing, it remains "fantastic", which is then decomposed into the following set of 6-character shingles:

$$\{\texttt{fantas, antast, ntasti, tastic}\}$$

This approach makes the similarity measure robust against minor variations in the text. For instance, if a user makes a typo or changes a single character, only the shingles overlapping that specific character will change, while the rest of the set remains identical. This allows the algorithm to detect near-duplicates effectively even when the reviews contain slight differences in spelling or formatting.

This preprocessing ensures that the dataset is suitable for shingling and scalable similarity detection using MinHash and LSH.

4.4 Hashing and Optimization

Storing shingles as strings is memory-intensive. To optimize storage and comparison speed:

4.4.1 Integer Hashing

Each string shingle was hashed to a 32-bit unsigned integer using the CRC32 algorithm(binascii.crc32).

4.4.2 Shingle Capping

To prevent exceptionally long reviews from consuming excessive memory during the MinHash signature generation, the number of shingles per document was capped at 2,000. If a document contained more than 2,000 unique shingles, only the first 2,000 were retained.

This preprocessing converted the raw text into a lightweight, numerical format ready for MinHashing.

5 Methodology: Similarity Detection

5.1 MinHashing

MinHashing is a probabilistic similarity estimation technique used to reduce the dimensionality of high-dimensional datasets while preserving the Jaccard similarity between them. Instead of storing the full set of k-shingles for every document—which requires significant memory—MinHash generates a fixed-length signature for each document. The fundamental property of this algorithm is that the probability of the MinHash signatures of two documents matching is equal to the Jaccard similarity of their underlying shingle sets.

In this project, I implemented **Vectorized MinHashing** with $n = 100$ hash functions. The process involved the following steps:

1. Generation of 100 pairs of random coefficients (a, b) to create independent hash functions of the form $h(x) = (ax + b) \pmod{p}$.
2. Calculation of the minimum hash value across all shingles for every document using vectorized operations.
3. Creation of a dense signature vector of length 100 for every review, drastically reducing the storage requirement compared to sparse shingle sets.

5.2 Locality Sensitive Hashing (LSH)

While MinHash reduces the dimensionality of documents, comparing every MinHash signature against every other signature would still require quadratic time complexity. Locality Sensitive Hashing (LSH) addresses this issue by using a banding technique to efficiently identify only those document pairs that are likely to be similar, referred to as *candidate pairs*.

The MinHash signature matrix is divided into multiple bands, with each band consisting of a fixed number of rows. Two documents are considered a candidate pair if their signatures match exactly in at least one band. This approach significantly reduces the number of comparisons required while retaining most truly similar pairs.

In this implementation, LSH was configured with the following parameters:

- **Bands (b):** 10
- **Rows per Band (r):** 10

This configuration results in a total of 100 hash functions and was chosen to make the candidate selection process conservative, prioritizing high-precision matches. Exact Jaccard similarity was then computed for all candidate pairs to verify similarity and filter out false positives.

5.3 LSH Candidate Generation Results

The application of Locality Sensitive Hashing (LSH) with the selected parameters ($b = 10, r = 10$) effectively filtered the vast search space. The algorithm identified **34,919 candidate pairs**.

5.4 Similarity Evaluation (Jaccard Similarity)

Once candidate pairs were identified using LSH, the final step was to verify their actual similarity by computing the exact Jaccard similarity score. Given two documents represented by shingle sets A and B, the Jaccard similarity is defined as:

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

Exact Jaccard similarity was computed for all candidate pairs. A similarity threshold of 0.55 was applied to determine which pairs should be considered truly similar. Pairs with a score below this threshold were discarded as false positives, while pairs with a score greater than or equal to 0.55 were retained as verified similar reviews.

This threshold was chosen to balance precision and recall, allowing the detection of near-duplicate reviews that share substantial lexical overlap while filtering out pairs with weak or incidental similarity. In practice, this value effectively captured reviews with minor edits, formatting differences, or small variations in wording without introducing excessive noise.

6 Results & Deduplication

6.1 Identification of Similar Pairs

Upon completing the MinHash and LSH pipeline and filtering the candidate pairs with a strict Jaccard similarity threshold of **0.55**, the algorithm successfully identified a significant volume of redundant content.

- **Total Verified Pairs:** The system detected **34,914** pairs of reviews that met or exceeded the similarity threshold.
- **High Similarity:** As observed in the sample output, the top pairs consistently exhibited a Jaccard similarity of **1.0**, indicating that a large portion of these matches are exact text duplicates.

6.2 Deduplication Strategy

To clean the dataset, a deterministic deduplication strategy was applied. For every identified pair (Doc_A, Doc_B) where $ID_A < ID_B$, the document with the higher ID (Doc_B) was flagged for removal. This ensures that for any cluster of duplicate reviews, one unique instance is preserved while all redundant copies are discarded.

6.3 Quantitative Impact

The deduplication process effectively reduced the dataset size by removing redundant entries. Note that the count of removed documents is lower than the count of pairs because a single duplicate document may be paired with multiple other documents within a duplicate cluster. The final statistics are presented in Table 2.

Metric	**Count**	**Description**
Original Documents	301,030	Size of the working sample before processing
Verified Similar Pairs	34,914	Pairs identified with Jaccard ≥ 0.55
Removed Duplicates	24,663	Unique documents removed from the dataset
Cleaned Documents	**276,367**	Final count of unique documents retained

Table 2: Summary of Deduplication Results

This resulted in a dataset reduction of approximately **8.2%**, significantly improving data quality for downstream analysis.

6.4 Distribution of Jaccard Similarities

To better understand the nature of the similarity between the detected book reviews, we analyzed the distribution of Jaccard similarity scores for all verified pairs (where similarity ≥ 0.55).

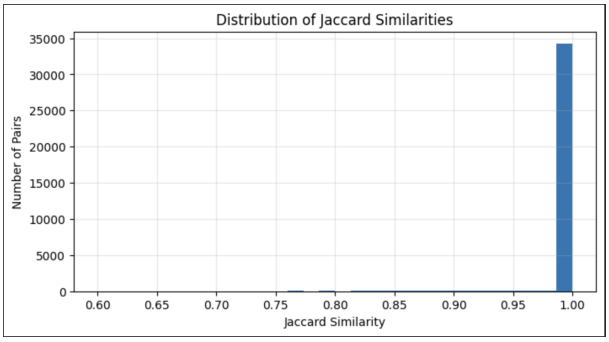

Figure 6: Boxplot of actual Jaccard similarity scores for matched pairs.

The histogram of these scores reveals the following insights:

- **Concentration at 1.0:** A significant portion of the pairs have a similarity score of exactly **1.0**. This confirms that the dataset contains a large number of **exact duplicates**—reviews that are identical in content but possess different unique IDs.
- **Near-Duplicates:** The distribution also shows pairs with scores between **0.55 and 1.0**, representing **near-duplicates**. These are likely reviews that have been slightly modified, edited, or have minor formatting differences.

This distribution validates the effectiveness of the chosen threshold (0.55), as it captures both identical copies and substantially similar content while excluding low-similarity noise.

7 Qualitative Analysis of Near-Duplicates

To validate the robustness of the similarity detection pipeline, I specifically analyzed "near-duplicate" pairs—defined as pairs with a Jaccard similarity score between **0.70 and 1.0**. These represent reviews that are highly similar but strictly non-identical.

Types of Variations Detected An inspection of the text content for these pairs revealed several common patterns of variation that exact deduplication methods would fail to catch:

- **Capitalization Differences:**
  - *Example:* Pair 68719598945 vs 68719598946 (Jaccard ≈ 0.975)
  - *Variation:* One review uses "ouidad" (lowercase) while the other uses "Ouidad" (capitalized). The content remains semantically identical.
- **Typos and Minor Edits:**
  - *Example:* Pair 60129599883 vs 60129599884 (Jaccard ≈ 0.936)
  - *Variation:* The phrase "best books i ever read" versus "best books ever read." The omission of the single letter 'i' reduces the similarity score but represents the same review.
- **Formatting and HTML Artifacts:** Several pairs exhibited differences only in special characters or HTML entity encoding (e.g., `&quot;` vs. standard quote marks), which MinHash successfully grouped together despite the raw string mismatch.

Table 3: Complete List of Near-Duplicate Review Pairs ($0.7 \leq$ Jaccard < 1.0)

Doc ID 1	Doc ID 2	Sim.	Review 1	Review 2
17179892272	25769876956	0.8032	This was a different kind of book for me, and I found it a challenge to finish it. It was heart breaking in many ways...	This was an enjoyable book, one to relax with. The plot was easy to follow and the characters were true to life. If y...
25769883932	25769893124	0.7774	"""On Liberty"" is one of the most important books on political thought of the nineteenth century. Fortunately for th...	"This Kindle edition has an introduction written more than a century ago that offers no insight into Mill's essay and...
25769868314	42949783243	0.9715	I just read this book for an Honors English class, and I got exactly what I expected. Lewis Carroll has an unique wri...	I just read this book for an Honors English class, and I got exactly what I expected. Lewis Carroll has an unique wri...
111025	51539720975	0.8930	This book is one of my favorites of all time. Many people dislike it or don't like it as much when compared to Pride ...	This book is one of my favorites of all time. Many people dislike it or don't like it as much when compared to Pride ...
34359747698	51539740274	0.8768	Easy to get on my Kindle and to transport it everywhere. Rapid download to all my connected electronics. And you cann...	Easy to get on my Kindle and to transport it everywhere. Rapid download to all my connected electronics. And you cann...
60129599883	60129599884	0.9358	I am not a big fan of books but,when I read "Animal Farm" for school it was one of the best books i ever re...	I am not a big fan of books but,when I read "Animal Farm" for school it was one of the best books ever read...
42949788311	68719500036	0.7931	stocking up on freebies... .it was a freebie. it was a paperless book. why not take advantage of it right? not like W...	.it was a freebie. it was a paperless book. why not take advantage of it right? not like WHOA! book!
68719598945	68719598946	0.9752	The authors of this book is ouidad and Jennifer Schonbrunn kinkle.Ouidad is known as the "Queen of Curl" wi...	The authors of this book is Ouidad and Jennifer Schonbrunn kinkle.Ouidad is known as the "Queen of Curl" wi...

Continued on next page

Table 3 – continued from previous page

Doc ID 1	Doc ID 2	Sim.	Review 1	Review 2
60129626168	68719537356	0.7402	Jane Eyre is a classic novel from Charlotte Bronte and this unabridged audio recording is highly recommended to peopl...	Emily Bronte's Wuthering Heights is a classic novel from Charlotte Bronte and this unabridged audio recording is high...
68719544860	77309473485	0.8864	Anyone who glances upon this text has read or is about to read this book. I have read this book and I can not express...	Anyone who glances upon this text has read or is about to read Great Expectations. I have read this book and I can no...

8 Discussion and Analysis of Results

8.1 Effectiveness of LSH-based Similarity Detection

The results show that the MinHash and LSH pipeline worked well for identifying similar book reviews at scale. Instead of comparing every review with every other review, which would have been too slow for a dataset of this size, the system was able to efficiently narrow down the search to a much smaller set of likely matches.

Out of 301,030 reviews, the pipeline identified 34,914 pairs with a Jaccard similarity of at least 0.55. Computing the exact Jaccard similarity for these candidate pairs confirmed that the matches produced by LSH were generally high quality. Many of the pairs had a similarity score of 1.0, meaning they were exact duplicates, while others had scores between 0.70 and 0.99. These near-duplicate reviews usually differed only by small changes such as typos, capitalization differences, or minor formatting issues.

After identifying similar pairs, a deterministic deduplication strategy was used to remove redundant reviews. This resulted in the removal of 24,663 reviews, reducing the dataset size by approximately 8.2%. Overall, these results show that combining MinHashing with LSH is an effective and scalable approach for detecting duplicate and near-duplicate text in large datasets.

9 Conclusion

The goal of this project was to detect similar book reviews in a large dataset using scalable methods, without relying on expensive exhaustive comparisons. By using MinHashing and Locality Sensitive Hashing, I was able to efficiently identify both duplicate and near-duplicate reviews in a dataset containing over 300,000 entries. The results demonstrate that probabilistic techniques can be used to clean large text datasets while maintaining good accuracy. Removing redundant reviews helped improve the overall quality of the data and made it more suitable for further analysis. In future work, this approach could be improved by experimenting with different similarity thresholds, shingle sizes, or by incorporating semantic similarity methods to capture deeper meaning beyond surface-level text similarity. Overall, this project shows that MinHash and LSH are practical and effective tools for large-scale similarity detection.