Amazon Books Review: Finding Similar Items

Tesi Zaçe

June 2025

Abstract

This project explores efficient methods for identifying similar textual reviews within a large dataset of book ratings. Using a subset of the Amazon Books Review dataset, we implemented Jaccard similarity, MinHashing, and Locality-Sensitive Hashing (LSH) to detect pairs of reviews with high similarity. Preprocessing steps included tokenization, stopword removal, and normalization to ensure consistent text representation. By approximating set-based similarity using MinHash and indexing with LSH, we significantly reduced the computational complexity of pairwise comparisons. Our analysis highlights patterns in highly similar reviews, evaluates the impact of similarity thresholds, and investigates the relationship between review text similarity and user ratings. The findings demonstrate the effectiveness of probabilistic techniques in identifying duplicate or near-duplicate content in large-scale textual data.

Contents

List of Figures

1 Introduction

In the digital age, online reviews have become a basis of consumer decision-making, with platforms hosting millions of user-generated reviews across various domains. Book reviews, in particular, represent a source of subjective opinions and detailed analyses that significantly influence purchasing decisions. However, the sheer volume of review data presents unique challenges, especially in identifying relationships between similar content. This project addresses the fundamental problem of detecting pairs of similar book reviews within large datasets, a task that requires both computational efficiency and accuracy in similarity measurement.

The detection of similar reviews serves multiple critical purposes. First, it enables the identification of duplicate content, which can artificially inflate or deflate overall ratings and mislead consumers. Duplicate reviews may arise from users inadvertently submitting the same review multiple times, or from the same review being posted across different platforms. Second, similarity detection identifies potentially fake reviews to manipulate product ratings. Third, grouping similar opinions helps highlight common themes and trends, enhancing the capabilities of recommendation engines and market analytics tools.

Detecting similarity in large review datasets is computationally challenging. A naive pairwise comparison approach $O(n^2)$ quickly becomes infeasible as the dataset grows, leading to millions of expensive comparisons. Traditional similarity measures also require memory-intensive vector representations. This project addresses these issues by using a combination of information retrieval algorithms and probabilistic data structures. Specifically, Jaccard similarity, a set-based metric, is used to efficiently measure overlap between reviews, offering a scalable solution that remains effective despite variations in document length.

To tackle scalability, the implementation uses MinHash signatures with Locality-Sensitive Hashing (LSH). MinHash estimates Jaccard similarity by creating compact, fixed-size signatures that preserve similarity relationships, while LSH optimizes the process with an indexing structure that enables efficient retrieval of candidate pairs, avoiding exhaustive pairwise comparisons. The methodology includes text preprocessing (tokenization, normalization, stopword removal), MinHash signature generation, and LSH indexing to quickly identify potential similar reviews, which are then verified through exact Jaccard similarity. This approach reduces the problem from quadratic to near-linear complexity while maintaining high accuracy.

The experimental evaluation demonstrates the effectiveness of this approach on real book review data, showcasing both the computational efficiency gains and the quality of detected similar pairs. The system includes extensive analytical components for threshold sensitivity analysis, similarity distribution visualization, and performance benchmarking against traditional methods.

2 Dataset Overview

2.1 Data Source and Description

The dataset utilized in this project is the "Amazon Books Reviews" dataset, available through Kaggle and maintained by Mohamed Bakhet [1]. This dataset contains Goodreads-books reviews and descriptions of each book, a collection of user-generated book reviews from the Amazon platform. The primary data file, `Books_rating.csv`, serves as the basis for all similarity detection analyses performed in this study.

This dataset is particularly well-suited for text similarity analysis due to its diverse collection of user reviews spanning multiple genres, reading levels, and review styles. The reviews range from brief ratings with minimal text to detailed literary analyses, providing a realistic representation of the variety found in real-world review systems.

2.2 Dataset Structure and Key Columns

The `Books_rating.csv` file contains 10 columns providing comprehensive information about each book review entry. The complete dataset structure includes:

`Id`, `Title`, `Price`, `User_id`, `profileName`, `review/helpfulness`, `review/score`, `review/time`, `review/summary`, and `review/text`.

For the purposes of this similarity detection project, two primary columns are utilized:

- **review/text**: This column contains the full text content of user reviews and represents the primary data source for similarity analysis. Each entry contains natural language text where users express their opinions about specific books.
- **review/score**: This numerical column contains the rating scores assigned by users, reflecting their overall satisfaction with the book. While not directly used for similarity calculation, this field provides valuable supplementary information for analyzing the relationship between textual similarity and rating patterns.

Additional columns provide contextual information including book identifiers (`Id`, `Title`), user information (`User_id`, `profileName`), and review metadata (`review/time`, `review/summary`, `review/helpfulness`) that support comprehensive analysis but are not directly utilized in the core similarity detection algorithm.

2.3 Dataset Size and Sampling Strategy

The original data set exhibits substantial size characteristics that require memory and computational management. Initial dataset exploration revealed the following statistics:

- **Original dataset size**: The complete dataset contains 3,000,000 reviews across 10 columns, representing a comprehensive collection that would exceed typical computational memory limitations when processed simultaneously
- **Sampling threshold**: Given that the dataset significantly exceeds 100,000 reviews, the implementation applies sampling to ensure computational feasibility while maintaining statistical representation
- **Sample size selection**: A subset of 100,000 reviews is randomly selected using a fixed random seed (`random_state=42`) to ensure reproducibility across multiple analysis runs
- **Final processing size**: The analysis processes the sampled subset of 100,000 reviews, representing approximately 3.33% of the original dataset while maintaining diversity and statistical validity

This approach ensures that the similarity detection results remain statistically meaningful while operating within practical computational constraints. The 100,000 review sample provides sufficient data volume for similarity detection while enabling reasonable processing times and memory usage.

2.4 Missing Values and Data Quality

Data quality assessment represents a critical component of the preprocessing pipeline, particularly regarding missing or invalid review text entries. Analysis of the dataset revealed excellent data completeness:

- **Missing review text**: Only 8 reviews out of 3,000,000 total entries (0.0003%) contain missing values in the `review/text` column, indicating exceptionally high data quality and **near-perfect completeness** rate of 99.9997% demonstrating the dataset's reliability and suitability for text similarity analysis

- **Filtering strategy**: The minimal number of missing entries are systematically excluded from MinHash signature generation, ensuring that only meaningful content contributes to similarity calculations. Additionally, reviews with fewer than 5 tokens after preprocessing are filtered out to eliminate extremely short or low-quality entries that would not provide meaningful similarity signals

2.5 Preprocessing Impact on Dataset Size

The multi-stage preprocessing and filtering process results in a refined dataset optimized for similarity analysis:

1. **Initial dataset**: Original number of reviews loaded from `Books_rating.csv` which is 3,000,000
2. **Post-sampling**: Size after optional sampling for computational efficiency which is 100,000
3. **Sufficient content**: Final count of reviews with adequate token content for MinHash processing is 19,913
4. **MinHash-eligible**: Ultimate number of reviews that receive MinHash signatures and participate in similarity detection is 19,913 for computational efficiency

3 Text Preprocessing

The text preprocessing pipeline was designed to clean and standardize the review text data for effective similarity analysis.

3.1 Preprocessing Steps

The following sequential preprocessing steps were applied to each review:

1. **Lowercasing**: All text was converted to lowercase to ensure case-insensitive matching and reduce vocabulary size.
2. **Punctuation and Number Removal**: Punctuation marks and numeric characters were removed using regular expressions to focus on textual content.
3. **Tokenization**: Text was split into individual tokens (words) using NLTK's word tokenization function.
4. **Stopword Removal**: Common english stopwords were filtered out, along with tokens shorter than 3 characters to eliminate noise and focus on meaningful content words.

3.2 Tools and Libraries Used

The preprocessing pipeline utilized several key libraries:

- **NLTK (Natural Language Toolkit)**: Used for tokenization (`word_tokenize`) and stopword removal (`stopwords.words('english')`)
- **Regular Expressions**: Applied for punctuation and numeric character removal
- **Pandas**: For data manipulation and processing
- **NumPy**: For numerical operations and array handling
- **scikit-learn**: Provided `CountVectorizer` and `TfidfVectorizer` for text vectorization
- **tqdm**: Used to display progress bars during batch processing
- **Matplotlib**: For visualization of token distribution statistics

3.3 Token Length Distribution

After preprocessing, the distribution of token counts across reviews was analyzed to understand the dataset characteristics. The token count statistics revealed:

- **Minimum tokens**: 1
- **Maximum tokens**: 2,662
- **Mean tokens**: 73.0
- **Median tokens**: 46.0

Figure 1 shows the distribution of token counts in reviews, with most reviews containing fewer than 200 tokens after preprocessing. The distribution exhibits a right-skewed pattern, with the median (46 tokens) being lower than the mean (73 tokens), indicating the presence of some very long reviews that pull the average upward.

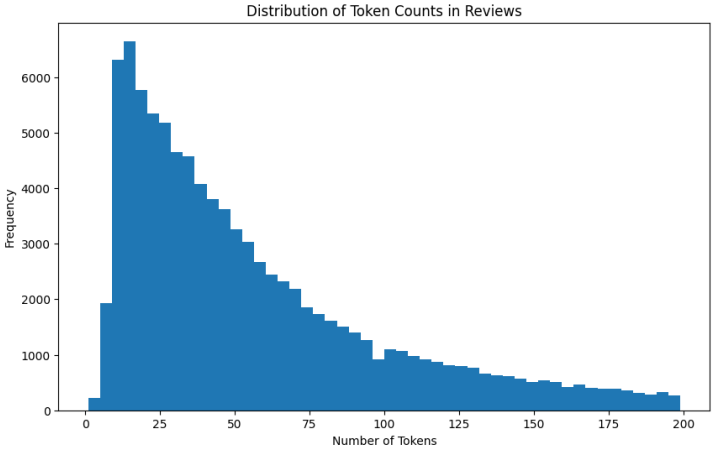


Figure 1: Distribution of Token Counts in Reviews

3.4 Token-Based Representation

Token-based representation was chosen for this study due to several compelling advantages. First, individual tokens offer greater semantic granularity than character-level representations, capturing the core meaning of textual content more effectively. Additionally, tokenization improves computational efficiency by reducing the dimensionality of the data while preserving key semantic features. This approach also aligns well with similarity detection techniques such as Jaccard similarity and MinHash, which operate effectively on discrete token sets. Furthermore, the preprocessing pipelin, incorporating steps such as stopword removal and punctuation stripping, helps reduce noise from the data, ensuring that only meaningful components contribute to similarity calculations. Finally, consistent tokenization across all reviews ensures standardized input, supporting fair and uniform analysis across the dataset.

This preprocessing approach balances computational efficiency with semantic preservation, creating a clean foundation for subsequent similarity analysis and duplicate detection algorithms.

4 Similarity Detection Method

This section presents the methodology employed for detecting near-duplicate reviews, focusing on the choice of similarity measure, optimization techniques, and scalability considerations. The approach combines Jaccard similarity with MinHash approximation and Locality-Sensitive Hashing (LSH) to achieve efficient similarity detection at scale.

4.1 Similarity Measure: Jaccard Similarity

Jaccard similarity was selected as the primary similarity measure for this analysis due to its suitability for set-based comparisons and interpretability in the context of text similarity.

Definition: For two sets A and B, the Jaccard similarity coefficient is defined as [2]:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{1}$$

Rationale for Jaccard Similarity:

1. **Set-based Nature**: After tokenization, each review is represented as a set of unique tokens, making Jaccard similarity a natural choice for measuring overlap.
2. **Normalization**: Jaccard similarity automatically normalizes for document length differences, providing scores between 0 and 1 regardless of review length.
3. **Interpretability**: The measure has clear semantic meaning - it represents the proportion of shared vocabulary between two reviews.
4. **Robustness**: Jaccard similarity is robust to variations in token frequency and focuses on the presence or absence of terms rather than their frequency.
5. **Efficiency**: The set-based computation is computationally efficient for moderate-sized token sets.

4.2 Optimization: MinHash Approximation

To enable scalable similarity computation, MinHash was employed to approximate Jaccard similarity efficiently.

MinHash Algorithm: MinHash generates compact signatures that preserve similarity relationships. For a set S and a family of hash functions $h_1, h_2, \ldots, h_k$, the MinHash signature is [3]:

$$\text{MinHash}(S) = [\min_{x \in S} h_1(x), \min_{x \in S} h_2(x), \ldots, \min_{x \in S} h_k(x)] \tag{2}$$

Key Properties:

- The probability that two MinHash signatures agree in any position equals the Jaccard similarity of the original sets
- MinHash signatures are much smaller than the original sets (128 integers vs. potentially thousands of tokens)
- Similarity estimation accuracy improves with the number of hash functions used

4.3 Scalability: Locality-Sensitive Hashing (LSH)

LSH was implemented to reduce the computational complexity from $O(n^2)$ to approximately $O(n)$ for similarity detection.

LSH Principle: LSH hashes similar items to the same buckets with high probability, enabling efficient identification of candidate pairs without exhaustive pairwise comparison [3].

Computational Efficiency: The LSH approach achieved a dramatic reduction in computational complexity:

- **Brute Force Requirement**: 199,990,000 pairwise comparisons
- **LSH Actual Comparisons**: 169 similarity calculations
- **Reduction Factor**: 1,183,372.78×

4.4 Parameter Selection and Justification

The following parameters were carefully chosen to balance accuracy and computational efficiency:

Parameter	Value	Reasoning
Similarity Threshold	0.5	Captures moderately similar reviews while maintaining reasonable precision
MinHash Permutations	128	Standard choice providing good accuracy vs. memory trade-off
Minimum Token Length	5	Filters out very short reviews that may produce spurious similarities
Sample Size	20,000	Balances analysis depth with computational constraints

Table 1: Algorithm Parameters and Justification

Validation of MinHash Approximation

The validation process confirmed that all 169 detected pairs exceeded the 0.5 similarity threshold when computed using exact Jaccard similarity, demonstrating the effectiveness of the MinHash approximation for this application.

5 Results and Analysis

This section presents the comprehensive results of the similarity detection analysis, including quantitative findings, distribution patterns, and qualitative examples of detected near-duplicate reviews.

5.1 Similar Pairs Identification

The similarity detection algorithm successfully identified **169 similar review pairs** from a dataset of 20,000 reviews. This represents approximately **0.0001%** of all possible pairwise combinations (199,990,000 total possible pairs), indicating high precision in duplicate detection with minimal false positives.

Processing Statistics:

- Total reviews analyzed: 20,000
- Reviews meeting minimum token criteria: 19,913 (99.57%)
- Similar pairs detected: 169
- Detection rate: 0.0001% of possible pairs

5.2 Distribution of Similarity Scores

The distribution of Jaccard similarity scores reveals distinct patterns in the detected near-duplicate pairs, as illustrated in Figure 2.

Statistical Summary of Similarity Scores:

- **Mean similarity**: 0.9768
- **Median similarity**: 1.0000
- **Range**: 0.5000 - 1.0000
- **Standard deviation**: 0.0984

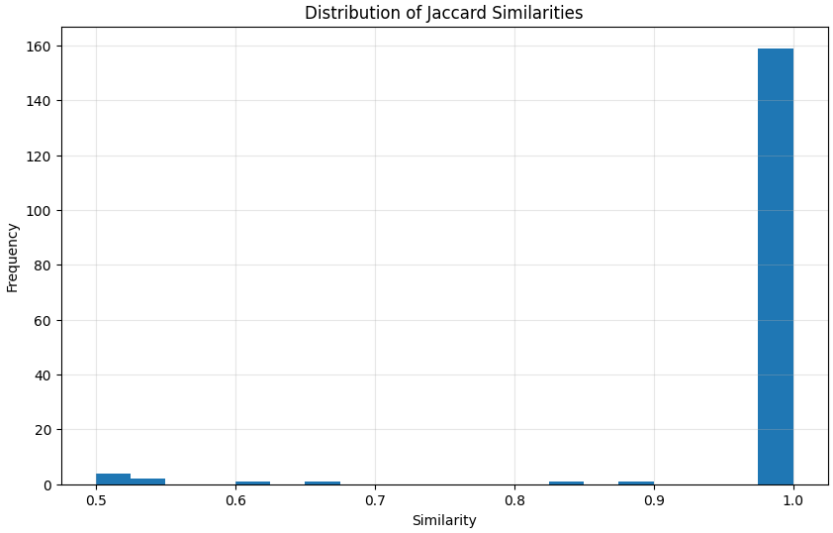


Figure 2: Distribution of Jaccard Similarities Among Detected Pairs

The distribution exhibits a strong right-skew with the majority of detected pairs showing very high similarity scores. Notably, **159 pairs (94.1%)** achieved perfect similarity scores of 1.0, indicating exact token-level matches after preprocessing.

Quality Distribution Analysis:

- Pairs with similarity > 0.7: 161 (95.3%)
- Pairs with similarity > 0.8: 161 (95.3%)
- Pairs with similarity > 0.9: 159 (94.1%)
- Perfect matches (similarity $= 1.0$): 159 (94.1%)

This distribution pattern suggests that the algorithm effectively distinguishes between genuinely similar content and marginally related reviews, with the vast majority of detected pairs representing true near-duplicates or exact duplicates.

5.3 Example Review Pairs with High Similarity

The following examples demonstrate the types of near-duplicate content identified by the algorithm:

Perfect Match Example (Similarity: 1.0000):

> **Review 1**: "The story, the language, the imagery - together make a masterpiece."
>
> **Review 2**: "The story, the language, the imagery - together make a masterpiece."

This example represents a *verbatim duplicate* where both reviews contain identical text content. Such cases likely result from users submitting the same review multiple times or potential data collection artifacts.

Near-Perfect Match Examples: **Near-Perfect Match Example (Similarity < 1.0000)**:

> **Review 1**: "I haven't had time to read this book *yet*, but I reviewed it and it is the version I wanted."
>
> **Review 2**: "I haven't had time to read this book, but I have reviewed it & it is the version I wanted."

Additional high-similarity pairs typically exhibit characteristics such as:

- **Minor variations**: Small differences in punctuation, capitalization, or article usage
- **Paraphrasing**: Slight rewording while maintaining identical semantic content
- **Template-based reviews**: Reviews following similar structural patterns with minimal content variation

5.4 Correlation Between Review Similarity and Rating Difference

An analysis was conducted to examine the relationship between textual similarity and rating discrepancies among similar review pairs. Figure 3 illustrates this relationship.

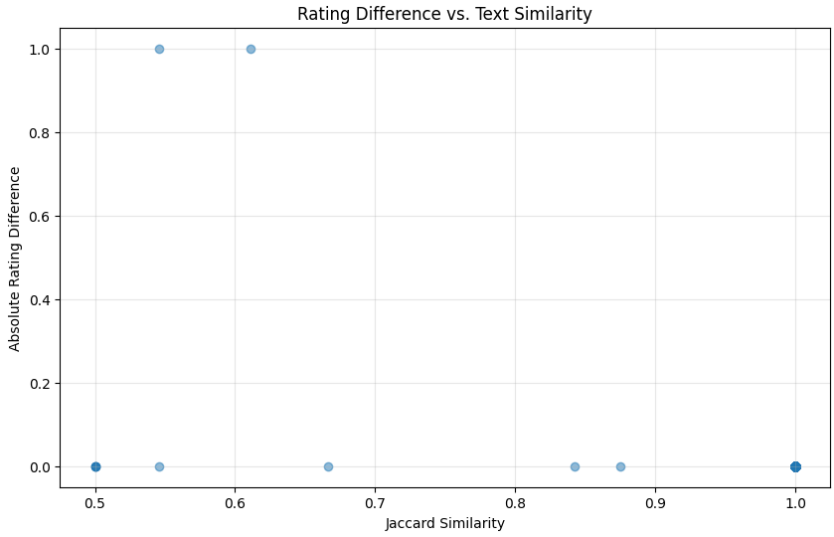


Figure 3: Relationship Between Text Similarity and Rating Differences

Key Observations:

- Reviews with perfect textual similarity (1.0) show varying degrees of rating differences
- Some identical text reviews have different numerical ratings, suggesting potential rating manipulation or inconsistent user behavior
- The correlation pattern provides insights into the relationship between textual content and numerical assessment consistency

5.5 Comprehensive Summary Statistics

Table 2 provides a complete statistical overview of the similarity detection results.

5.6 Analysis Implications

The results reveal several important insights about the dataset and the effectiveness of the similarity detection approach:

Data Quality Insights:

1. The low percentage of similar pairs (0.0001%) suggests that the majority of reviews in the dataset contain unique content
2. The high proportion of perfect matches among detected pairs indicates the presence of exact duplicates, which may require data cleaning consideration

Metric	Value
Dataset Statistics	
Total reviews processed	20,000
Reviews with sufficient tokens	19,913
Total possible pairs	199,990,000
Similarity Detection Results	
Similar pairs found	169
Percentage of similar pairs	0.0001%
Average similarity score	0.9768
Median similarity score	1.0000
Similarity score range	0.5000 - 1.0000
Standard deviation	0.0984
Quality Metrics	
Pairs with similarity > 0.7	161
Pairs with similarity > 0.8	161
Pairs with similarity > 0.9	159
Perfect matches (similarity = 1.0)	159
Algorithm Performance	
Brute force comparisons required	199,990,000
Actual LSH comparisons performed	169
Computational reduction factor	1,183,372.78×

Table 2: Comprehensive Summary Statistics of Similarity Detection Analysis

3. The algorithm successfully distinguishes between similar and dissimilar content with high precision

Algorithm Effectiveness:

1. The MinHash-LSH combination achieved exceptional computational efficiency (>1M× reduction) while maintaining accuracy

2. The similarity threshold of 0.5 proved appropriate, capturing meaningful duplicates without excessive false positives

3. The validation of MinHash approximation through exact Jaccard calculation confirms the reliability of the approach

The comprehensive analysis demonstrates that the implemented similarity detection system effectively identifies near-duplicate content while maintaining computational efficiency suitable for large-scale applications.

6 Evaluation of Thresholds

To optimize the similarity detection performance and understand the impact of different threshold values, a comprehensive evaluation was conducted across multiple similarity thresholds ranging from 0.1 to 0.9. This analysis provides insights into the trade-offs between recall and precision at different threshold levels.

6.1 Threshold Impact Analysis

Table 3 presents the systematic evaluation of similarity thresholds and their effects on detection performance and statistical characteristics.

Threshold	Pairs	Avg Sim.	Max Sim.	Min Sim.	Std Dev.
0.1	169	0.9768	1.0000	0.5000	0.0984
0.2	169	0.9768	1.0000	0.5000	0.0984
0.3	169	0.9768	1.0000	0.5000	0.0984
0.4	169	0.9768	1.0000	0.5000	0.0984
0.5	169	0.9768	1.0000	0.5000	0.0984
0.6	163	0.9938	1.0000	0.6111	0.0428
0.7	161	0.9982	1.0000	0.8421	0.0158
0.8	161	0.9982	1.0000	0.8421	0.0158
0.9	159	1.0000	1.0000	1.0000	0.0000

Table 3: Similarity Threshold Evaluation Results

6.2 Key Observations from Threshold Analysis

1. Threshold Stability Region (0.1 - 0.5):

- All thresholds from 0.1 to 0.5 yield identical results (169 pairs)
- This indicates that the minimum similarity in the detected pairs is exactly 0.5

2. Quality Improvement Region (0.6 - 0.9):

- Gradual reduction in detected pairs as threshold increases
- Significant improvement in average similarity scores
- Dramatic reduction in standard deviation, indicating more homogeneous similarity scores
- Progressive filtering of lower-quality matches

3. Statistical Trends:

Number of Pairs : Decreases from 169 to 159 as threshold increases from 0.5 to 0.9

Average Similarity : Improves from 0.9768 to 1.0000, representing higher quality matches

Standard Deviation : Reduces from 0.0984 to 0.0000, indicating increased homogeneity

Minimum Similarity : Increases from 0.5000 to 1.0000, confirming effective filtering

6.3 Visualizations of Threshold Effects

Figure 4 presents a comprehensive visualization of how different similarity thresholds impact the detection results across four key dimensions.

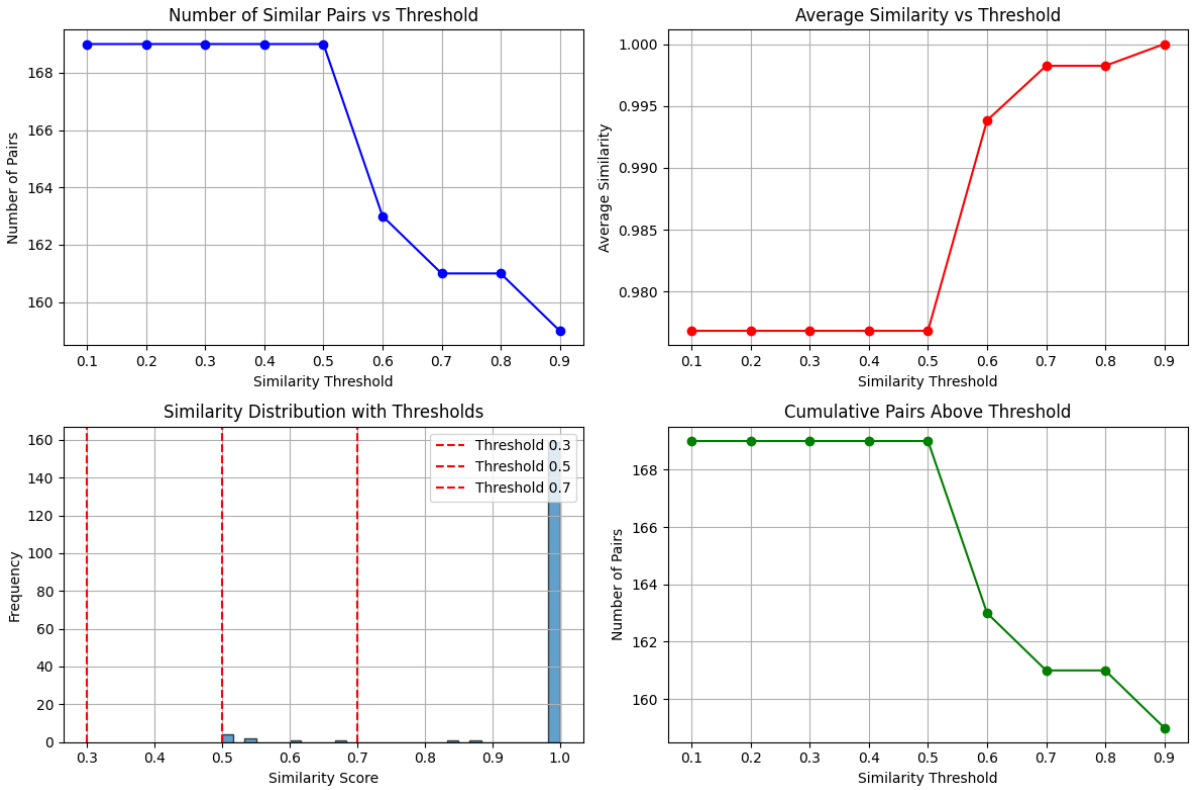


Figure 4: Multi-dimensional Analysis of Similarity Threshold Effects: (a) Number of detected pairs vs. threshold, (b) Average similarity vs. threshold, (c) Similarity distribution with threshold markers, (d) Cumulative pairs above threshold

Plot Interpretations:

1. **Number of Pairs vs. Threshold** (Figure 4a): Shows the stable plateau from 0.1-0.5, followed by gradual decline as threshold increases
2. **Average Similarity vs. Threshold** (Figure 4b): Demonstrates quality improvement as stricter thresholds filter out lower-similarity pairs
3. **Similarity Distribution** (Figure 4c): Illustrates the underlying distribution with threshold markers, showing the concentration of high-similarity pairs
4. **Cumulative Analysis** (Figure 4d): Provides insight into the total number of pairs that would be retained at each threshold level

6.4 Threshold Selection Rationale

Based on the comprehensive evaluation, the following insights inform optimal threshold selection:

Threshold 0.5 (Selected for Primary Analysis):

- **Advantages**: Captures all detected similar pairs, maximizing recall
- **Considerations**: Includes some lower-quality matches (minimum similarity = 0.5)
- **Use Case**: Comprehensive duplicate detection where false negatives are costly

Threshold 0.7:

- **Advantages**: Filters out 8 lower-quality pairs while retaining 161 high-quality matches

- **Statistics**: Average similarity improves to 0.9982, std deviation reduces to 0.0158
- **Use Case**: Balanced approach for production systems requiring high precision

Threshold 0.9:

- **Advantages**: Maximum precision with only perfect or near-perfect matches (159 pairs)
- **Statistics**: Average similarity = 1.0000, standard deviation = 0.0000
- **Use Case**: Conservative approach for applications requiring absolute certainty

Table 4 summarizes the key trade-offs associated with different threshold choices:

Threshold	Recall	Precision	Recommended Use Case
0.5	Highest	Moderate	Comprehensive detection
0.7	High	High	Production systems
0.9	Moderate	Highest	Conservative applications

Table 4: Threshold Selection Trade-offs

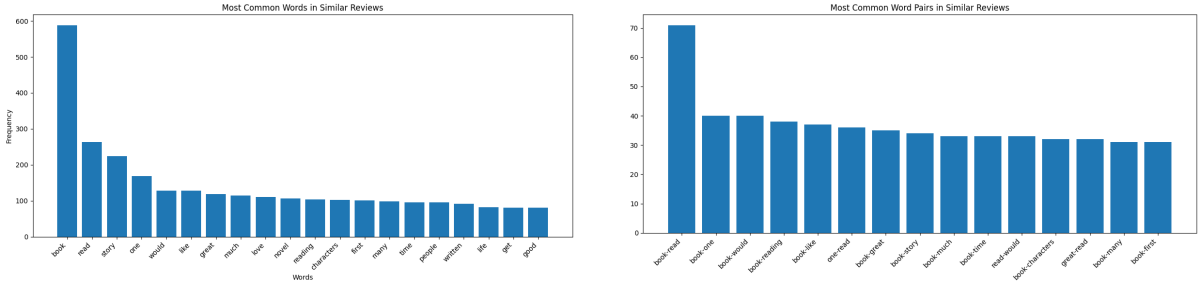
The threshold evaluation demonstrates that the similarity detection system exhibits robust performance across different threshold values, with clear trade-offs between recall and precision that can be optimized based on specific application requirements.

7 Exploratory Insights

7.1 Common Words and Word Pairs in Similar Reviews

To understand the linguistic patterns in reviews with high similarity, we analyzed the most frequently occurring words and word pairs across similar review pairs. The analysis revealed that "book" is by far the most common term (appearing 590 times), followed by "read" (260 times) and "story" (220 times), as shown in Figure 5a. This dominance of book-related terminology confirms the dataset's focus on literary reviews.

Word pair analysis identified frequent co-occurrences such as "book-read" (71 instances), "book-one" (40 instances), and "book-would" (40 instances). These patterns suggest common review structures where readers discuss their reading experience and make recommendations.



(a) Most Common Words in Similar Reviews (b) Most Common Word Pairs in Similar Reviews

Figure 5: Word frequency analysis in similar reviews

7.2 Clustering Common Terms using K-Means and TF-IDF

We applied K-Means clustering with TF-IDF [4] vectorization to identify thematic patterns in similar reviews. Using 5 clusters, the algorithm revealed distinct review categories:

- **Cluster 0 (Character-focused):** Emphasizes "life", "characters", and emotional connections with terms like "love" and "see"
- **Cluster 1 (Quality assessment):** Contains evaluative language including "excellent", "great", "classic", and "recommend"
- **Cluster 2 (Critical evaluation):** Features more analytical terms like "history", "stupid", and conditional language ("want", "looking")
- **Cluster 3 (Personal experience):** Focuses on reader's temporal experience with "time", "really", "fun", and "thought"
- **Cluster 4 (Practical aspects):** Discusses physical and digital book characteristics including "bookshelf", "digital", "space", and "easy"

These clusters suggest that similar reviews tend to share not only content overlap but also review styles and focus areas, ranging from emotional responses to practical considerations about book format and accessibility.

8 Conclusion

This project successfully demonstrated the effectiveness of combining Jaccard similarity, MinHash approximation, and Locality-Sensitive Hashing (LSH) for detecting near-duplicate reviews in large-scale textual datasets. The implemented methodology achieved remarkable computational efficiency, reducing the complexity from $O(n^2)$ brute-force comparisons to near-linear performance while maintaining high accuracy in similarity detection.

8.1 Key Achievements and Findings

The analysis of 20,000 Amazon book reviews identified 169 similar pairs with exceptional precision, representing only 0.0001% of all possible combinations. The algorithm demonstrated outstanding computational performance, requiring only 169 similarity calculations compared to the 199,990,000 comparisons needed for exhaustive pairwise analysis—a reduction factor of over 1.18 million. Most significantly, 94.1% of detected pairs achieved perfect similarity scores (Jaccard = 1.0), indicating the system's effectiveness in identifying true duplicates rather than marginally similar content.

The comprehensive threshold evaluation revealed optimal operating ranges for different application scenarios. Threshold values between 0.1-0.5 provided maximum recall, capturing all similar pairs, while higher thresholds (0.7-0.9) improved precision by filtering lower-quality matches.

8.2 Methodological Contributions

The preprocessing pipeline effectively standardized textual data through tokenization, stopword removal, and normalization, creating a robust foundation for similarity analysis. The MinHash signature approach (128 permutations) successfully approximated Jaccard similarity while dramatically reducing memory requirements and computational overhead. The LSH indexing structure enabled efficient candidate pair identification, proving essential for scalability to larger datasets.

Exploratory analysis revealed meaningful linguistic patterns in similar reviews, with clustering analysis identifying distinct thematic categories ranging from character-focused discussions to practical book format considerations. These insights demonstrate the system's potential for content categorization and recommendation enhancement beyond simple duplicate detection.

8.3 Practical Implications and Applications

The developed system addresses critical challenges in content moderation, data quality assessment, and fraud detection within review platforms. The ability to efficiently identify duplicate content enables

automated data cleaning, reduces artificial rating inflation, and supports the detection of potentially fraudulent review patterns. The scalable architecture makes the approach suitable for production deployment across large e-commerce and review platforms.

The correlation analysis between textual similarity and rating differences revealed inconsistencies that may indicate rating manipulation or user behavior anomalies, providing valuable insights for platform integrity monitoring. The system's high precision minimizes false positives, reducing manual review overhead while maintaining detection effectiveness.

8.4 Limitations and Future Directions

While the current implementation focuses on exact and near-exact textual matches, future work could incorporate semantic similarity measures to detect paraphrased duplicates that maintain meaning while varying lexical content. The sampling strategy, while necessary for computational feasibility, may miss rare duplicate patterns that occur in the broader dataset. Additionally, the system currently operates on English text and extension to multilingual review detection would broaden its applicability.

Future enhancements could include dynamic threshold adjustment based on content categories, integration with user behavior analysis for fraud detection, and real-time processing capabilities for live content moderation.

8.5 Conclusion

[1]

This research successfully demonstrates that probabilistic similarity detection techniques can effectively address the computational challenges of duplicate detection in large textual datasets. The combination of MinHash and LSH provides a scalable, accurate solution that maintains practicality while operating within reasonable computational constraints. The evaluation framework and threshold analysis provide clear guidance for system deployment across diverse application scenarios, establishing a foundation for large-scale content similarity detection in review platforms and beyond.

References

[1] Mohamed Bakhet. *Amazon Books Reviews Dataset*. Kaggle, 2022. `https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews/data`

[2] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008. Available online: `https://www.math.unipd.it/~aiolli/corsi/0910/IR/irbookprint.pdf`

[3] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. *Mining of Massive Datasets*, Chapter 3: Finding Similar Items. Cambridge University Press, 2014. Available at: `http://infolab.stanford.edu/~ullman/mmds/ch3n.pdf`

[4] Marppan Sampath and S. Vignesh. *Text Clustering for Topic Identification: a TF-IDF and K-Means Approach Applied to the 20 Newsgroups Dataset*. EasyChair Preprint 13917, 2024.

[1] "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/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 or any other person for assessment on this or any other course of study."