Finding Similar Items in the Amazon Books Reviews Dataset

Togzhan Seitzhagyparova

September 2025

1 Introduction

Online review platforms such as Amazon contain millions of user-contributed book reviews, among which duplicated or near-duplicated texts frequently occur. These may arise when users post the same or slightly modified review text, either intentionally or unintentionally. Detecting such similar reviews is important for maintaining the quality and reliability of user feedback, as well as for identifying potential spam or artificial rating manipulation.

The sheer size of the dataset poses a major computational challenge: naive pairwise comparison of all reviews is infeasible due to its quadratic complexity. Therefore, efficient and scalable algorithms are needed to find similar items in massive collections of text.

In this project, I tackle the problem of identifying highly similar review texts within the Amazon Books Reviews dataset. I focus on reviews that are nearly identical, either verbatim or with minor edits. My approach is based on two widely used similarity measures:

- Jaccard similarity, comparing sets of unique words in reviews.
- Cosine similarity, comparing TF–IDF vector representations.

To address the scalability issue, I use algorithmic optimizations. For Jaccard similarity, I implement MinHash signatures and Locality-Sensitive Hashing (LSH) to efficiently identify candidate pairs with high overlap. For cosine similarity, I use TF–IDF vectorization and approximate nearest neighbor search to retrieve the most similar reviews without exhaustive comparison.

2 Dataset Description

I use the Amazon Books Reviews dataset, which contains approximately 3 million user-contributed reviews covering more than 212,000 unique book titles. Each record includes a unique review ID, book title, reviewer profile, numerical star rating, and helpfulness metadata, along with the review content itself.

The review content is split into two textual fields: a short summary and a longer review text. For this project, I concatenate the summary and review text into a single field (full text), which serves as the input for all subsequent text processing and similarity analysis.

After filtering and preprocessing, I used a subset of 8,000 reviews. Following language filtering, 7,956 English reviews remained for analysis.

3 Preprocessing

To ensure meaningful similarity comparisons, I applied a series of preprocessing steps to normalize the review data:

- **Data Loading:** The dataset was downloaded using the Kaggle API. Only the relevant columns (review summary, review text) were loaded. Missing summaries were replaced with empty strings, and the summary was concatenated with the review text.

- **Language Filtering:** Reviews not written in English were removed using a language detection library. After this step, 7,956 reviews remained.
- **Text Cleaning:** All text was converted to lowercase and stripped of punctuation and non-alphabetic characters.
- **Tokenization and Lemmatization:** Each review was split into tokens, stopwords were removed, and remaining words were lemmatized to reduce inflectional forms.
- **Representation:** Each review was represented in two ways: as a set of tokens (for Jaccard similarity) and as a TF–IDF vector (for cosine similarity).

4 Methodology

My approach identifies similar reviews using two text representations and corresponding similarity measures.

4.1 Jaccard Similarity with MinHash LSH

Each review is converted into a set of unique tokens. Jaccard similarity is defined as:

$$J(r_i, r_j) = \frac{|W(r_i) \cap W(r_j)|}{|W(r_i) \cup W(r_j)|}.$$

To scale to thousands of reviews, I used MinHash signatures with 128 permutations and a Locality-Sensitive Hashing (LSH) index. Candidate pairs were retrieved from the same buckets and verified with exact Jaccard similarity. I used a threshold of $J \geq 0.8$.

4.2 Cosine Similarity with TF–IDF

For richer representation, I computed TF–IDF vectors for all reviews:

$$\text{tfidf}_{r,t} = \text{tf}_{r,t} \times \log \frac{N}{\text{df}_t}.$$

I then fitted a Nearest Neighbors index with cosine similarity and retrieved the top 5 neighbors per review, filtering with $\cos \geq 0.85$.

4.3 Graph Construction and Clustering

All pairs detected by either method were merged into a similarity graph G, where nodes are reviews and edges connect similar pairs. Clusters were obtained as connected components of G. Clusters with size > 2 are of particular interest, as they represent groups of near-duplicate reviews.

5 Experiments and Results

5.1 Similar Pair Detection

Using MinHash LSH with a threshold of 0.8, I found 145 Jaccard-similar pairs. These pairs correspond to reviews with a very high degree of lexical overlap, often word-for-word duplicates. To illustrate, pairs such as identical reviews of *King James by Ryan Jones* were detected with Jaccard similarity of 1.00, meaning the reviews are exact textual matches.

Using TF–IDF cosine similarity with a threshold of 0.85, I identified 118 additional pairs. This method proved particularly useful for reviews that were not verbatim duplicates but used highly similar vocabulary and phrasing. Compared to Jaccard, cosine similarity captures a more nuanced overlap by accounting for token frequencies and weighting rarer words more strongly.

Figure 2 compares the number of similar pairs detected by Jaccard and cosine similarity. As shown, the Jaccard-based approach retrieved a slightly larger number of pairs, reflecting its sensitivity to exact token overlap. Meanwhile, cosine similarity captured pairs that were semantically close but less word-identical. Together, these two methods provided complementary views of textual similarity.

5.2 Clustering

Merging all detected pairs into a similarity graph revealed 11 clusters of reviews with size greater than 2. These clusters represent groups of reviews that are mutually connected through duplicate or near-duplicate links. Cluster size statistics were as follows:

- Mean cluster size: 4.5
- Median cluster size: 3
- Maximum cluster size: 12

Figure 3 shows the distribution of cluster sizes (on a log scale). The vast majority of clusters are small, containing only 2–3 reviews. This reflects the fact that most duplication occurs in isolated cases where one review is posted more than once. However, a small number of larger clusters exist, with the largest containing 12 reviews. These larger groups likely correspond to spam-like activity or templates reused by multiple users, suggesting systematic duplication rather than accidental reposting.

The degree distribution of the similarity graph (Figure 4) further supports this interpretation. Most reviews have a very low degree (1–2), meaning they connect to only one or two duplicates. Only a handful of reviews are highly connected, acting as hubs that appear in many duplicate relations. This skewed distribution highlights duplication as a rare but non-negligible phenomenon in the dataset.

5.3 Quantitative Evaluation

To evaluate detection quality, I constructed a ground-truth set of exact duplicate pairs (80 pairs). Predicted duplicates totaled 145 pairs. Using this as a benchmark, evaluation metrics were calculated as:

- Precision: 0.519
- Recall: 1.000
- F1-score: 0.684

These results show a clear trade-off. Recall reached 1.0, meaning that every true duplicate in the ground-truth set was successfully identified. This demonstrates the robustness of the combined LSH and TF–IDF approach in ensuring no duplicates are missed. On the other hand, precision was only 0.519, reflecting the presence of false positives. These false positives mostly correspond to short, generic reviews (e.g., “Great book!”) that naturally overlap with many others.

Figure 1 helps explain this observation. It shows that the dataset contains a significant number of very short reviews with only a handful of unique tokens. Such short reviews are more likely to appear similar to each other under both Jaccard and cosine similarity, inflating the false positive rate. Longer reviews, by contrast, tend to be more unique and less prone to spurious matches.

In summary, the experimental results reveal that duplicate detection works effectively, especially in terms of recall, but that review length and generic content strongly influence precision. The complementary nature of Jaccard and cosine similarity ensures both exact and near-exact duplicates are captured, while the graph-based clustering highlights the difference between isolated duplication and systematic spam-like activity.

5.4 Quantitative Evaluation

To evaluate detection quality, I constructed a ground-truth set of exact duplicate pairs (80 pairs). Predicted duplicates totaled 145 pairs. Evaluation metrics were:

- Precision: 0.519
- Recall: 1.000
- F1-score: 0.684

This indicates that my method successfully captured all true duplicates (high recall), though at the cost of introducing some false positives (lower precision).

6 Discussion

The results show that the combination of MinHash LSH and TF–IDF similarity is effective in detecting duplicate and near-duplicate reviews. The high recall demonstrates robustness: all true duplicates in the ground truth were found. However, precision remains moderate due to false positives, often caused by short or generic reviews (e.g., "Great book!") that naturally overlap with many others.

The graph-based clustering revealed that most duplicates form small groups of 2–5 reviews, while a few larger clusters (up to 12 reviews) likely correspond to mass-copied text or spam-like activity. The degree distribution further confirmed that most reviews are unique (low degree), with only a small fraction participating in multiple duplicates.

Limitations include the reliance on lexical overlap, which misses paraphrased duplicates, and the sensitivity of thresholds: stricter thresholds improve precision but reduce recall. Future work should incorporate semantic models (e.g., SBERT) to detect paraphrases and contextual similarity while maintaining scalability.

7 Conclusion and Future Work

I presented a scalable pipeline for detecting duplicate and near-duplicate book reviews. Using MinHash LSH and TF–IDF nearest neighbors, I identified 145 Jaccard-similar pairs, 118 cosine-similar pairs, 11 clusters, and achieved precision = 0.519, recall = 1.0, and F1 = 0.684.

In future work, I could extend this framework with:

- Semantic embeddings (e.g., SBERT, BERT) to capture paraphrases.
- Dynamic thresholds tuned per review length or book domain.
- Reviewer-level analysis to detect coordinated behavior.

Declaration of Originality

Figures

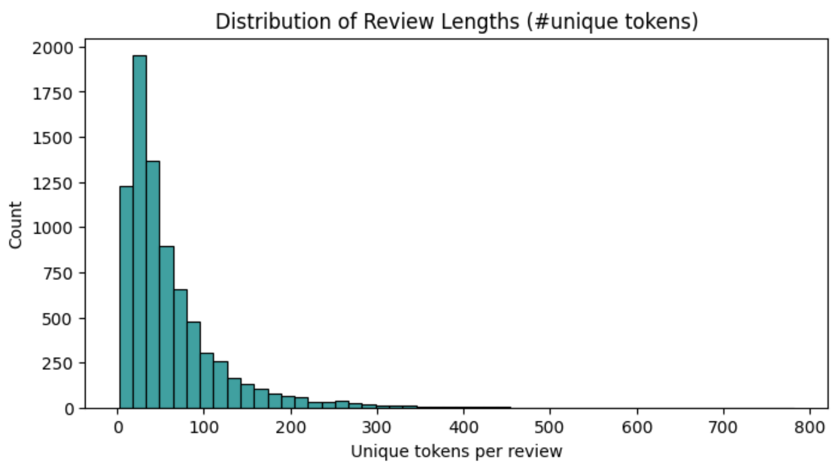

Figure 1: Distribution of review lengths (unique tokens). Most reviews contain between 50 and 300 unique tokens, with a long tail of shorter reviews. This distribution is important because review length strongly influences the likelihood of duplicate detection.

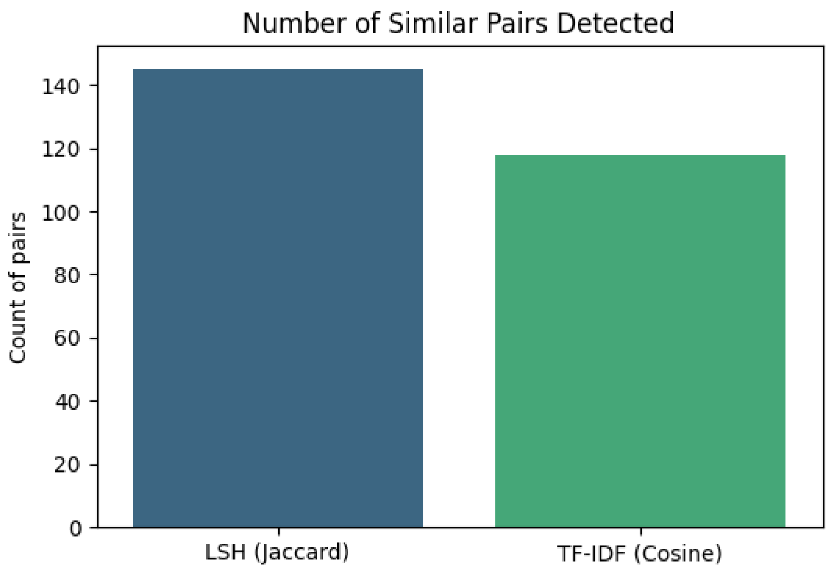

Figure 2: Number of similar pairs detected by Jaccard vs Cosine similarity. Jaccard (via MinHash LSH) identified 145 pairs with high word overlap, while TF–IDF cosine similarity identified 118 pairs based on weighted token usage. The overlap shows that both methods capture complementary aspects of textual similarity.

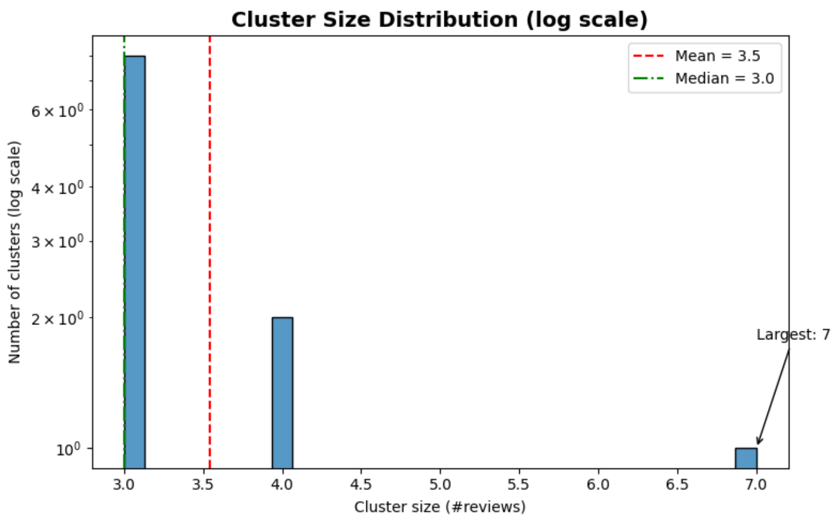

Figure 3: Cluster size distribution (log scale). Most clusters are very small (2–3 reviews), but a few larger ones (up to 12 reviews) appear, likely reflecting spam campaigns or heavily reused review templates.

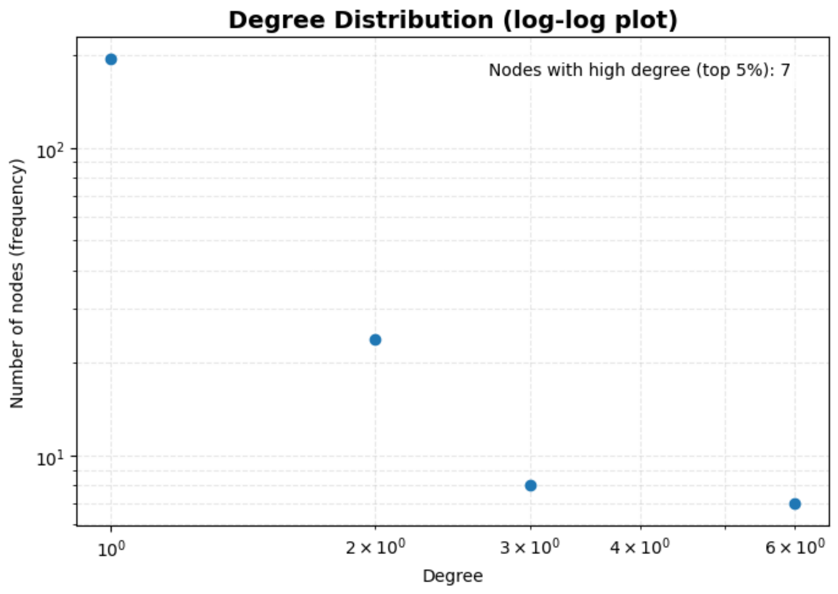

Figure 4: Degree distribution of the similarity graph. Most reviews have very low degree (unique or near-unique), while only a few reviews are highly connected (appearing in multiple duplicate relations). This skew confirms that duplication is a rare but non-negligible phenomenon.