Detecting Similar Book Reviews

Sara Miraglia & Sara Peri

July 11, 2025

Abstract

This report describes a methodology to detect pairs of similar book reviews by comparing their textual content. We process the full-text reviews from the `Books_rating.csv` dataset, apply two similarity measures—Jaccard and Cosine—and evaluate the most similar pairs. We discuss data organization, preprocessing, algorithmic details, scalability considerations, experimental setup, and analysis of results.

1 Introduction

In this project, we focus on pairs of Amazon book reviews that exhibit high textual overlap or semantic similarity. We compare a simple set-based Jaccard similarity with a TF–IDF-based Cosine similarity.

2 Dataset

The dataset was downloaded from Kaggle's "Amazon Book Reviews" repository[1] and is provided as a CSV file named `Books_rating.csv`. It contains raw Amazon book review records; Each record comprises the following fields:

- **Id**: unique identifier of the review.
- **Title**: title of the book being reviewed.
- **Price**: listed price of the book (may be null).
- **User_id**: anonymized identifier of the reviewer.
- **profileName**: name of the reviewer.
- **review/helpfulness**: fraction of users who found the review helpful.
- **rating**: star rating given by the reviewer.
- **review/time**: UNIX timestamp of the review.
- **review/summary**: short summary provided by the reviewer.
- **review_text**: full text of the review.

‘ For our project, we randomly shuffled the DataFrame (using a fixed seed of 42 for reproducibility), and selected the first `SUBSAMPLE_SIZE` rows as a representative subsample.
For our analysis, we focused exclusively on the textual content of the reviews, as contained in the `review/text` column (renamed internally as `review_text`). We also considered the associated `review/score` (renamed as `rating`) for potential filtering and analysis.

3 Data Organization

After loading and filtering the dataset, each review was assigned a unique numeric identifier (`id`) to facilitate pairwise comparisons. The key columns retained in the working DataFrame were:

- `id`: unique identifier for the review;
- `review_text`: original review content;
- `rating`: numerical score associated with the review;
- `cleaned`: a cleaned version of the review text (lowercased, punctuation removed, normalized whitespace);
- `shingles`: a list of k-character shingles (substrings) extracted from the cleaned text;
- `features`: the hashed vector representation of the shingle set, used for LSH (via `HashingTF`).

[1] `https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews`

Each row in the dataset thus represents a single review, enriched with both raw and processed versions of the text, as well as hashed feature vectors for similarity estimation.

After running the MinHashLSH similarity join, an additional table of candidate review pairs was constructed, with columns:

- `id1`, `id2`: identifiers of the two matched reviews;
- `review1`, `review2`: original texts of the two reviews;
- `rating1`, `rating2`: associated ratings;
- `jaccardSimilarity`: estimated Jaccard similarity between the hashed vectors.

‘

4 Pre-processing Techniques

A sequence of text pre-processing steps was applied to standardize and prepare the review texts for shingling and vectorization. The following techniques were implemented:

- **Null filtering**: all reviews with missing or empty text fields were discarded.
- **Minimum length filtering**: only reviews longer than 20 characters were retained to ensure a sufficient amount of content for comparison.
- **Text normalization**: each review was lowercased, punctuation was removed using regular expressions, and multiple whitespace characters were collapsed into a single space.
- **Shingling**: normalized review texts were tokenized into unique overlapping substrings (shingles) of fixed length $k = 5$ characters. This process captures local patterns and provides a more granular token-level representation.

The resulting sets of shingles were then used to construct high-dimensional sparse feature vectors using `HashingTF`, suitable for MinHashLSH similarity estimation. Each review thus became associated with a hashed binary vector that summarizes its content based on character-level features.

5 Implemented Algorithms

MinHash LSH and Jaccard Similarity

The core component of the system is based on the detection of syntactic similarity using the Jaccard index, a well-established metric for measuring the overlap between sets. In this case, character-level k-shingles ($k = 5$) are extracted from each cleaned review, producing a sparse set of tokens per document.

To scale the comparison of all review pairs, we apply the MinHash Locality Sensitive Hashing (LSH) technique, which approximates Jaccard similarity using hash signatures. The implementation relies on Spark's `MinHashLSH`, configured with 5 hash tables and 2^{18} features (hash buckets).

The approximate similarity join is performed between the dataset and itself. A pair of reviews (A, B) is retained if the estimated Jaccard similarity satisfies:

$$0.2 \leq J(A, B) \leq 0.75$$

where the lower bound selects sufficiently similar items, and the upper bound avoids near-duplicate or identical reviews.

TF-IDF Cosine Similarity (Optional Refinement)

To further filter the candidate pairs returned by LSH, a second-stage check based on semantic similarity is applied. Each review is transformed into a TF-IDF vector through a pipeline of tokenization, stopword removal, CountVectorizer and IDF weighting.

The cosine similarity is then computed on each candidate pair as:

$$\cos(\theta) = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

Only those pairs whose cosine similarity is at least 0.3 are retained in the final result set.

This additional semantic filtering step helps discard pairs that are lexically similar but semantically distant, improving overall precision. However, it is applied *only to Jaccard-matched pairs*, not as a standalone similarity measure.

Result Selection

The final dataset of similar reviews consists of all pairs that satisfy both:

- Jaccard similarity in $[0.2, 0.75]$
- Cosine similarity ≥ 0.3

This hybrid approach balances scalability, syntactic similarity, and semantic coherence..

6 Scalability and Performance

The entire implementation has been designed with scalability in mind, leveraging the Apache Spark framework for distributed data processing. All major steps – from reading the dataset to computing similarities – are implemented using Spark DataFrames and transformations, allowing automatic parallelism on multiple cores or across a cluster.

The preprocessing phase, including filtering, text normalization, and the generation of character shingles, is handled via Spark UDFs and transformations, which scale linearly with data size.

The core of the similarity detection relies on the `MinHashLSH` module provided by Spark MLlib, which efficiently approximates Jaccard similarity using locality-sensitive hashing. This avoids the need for an exhaustive pairwise comparison, which would have quadratic complexity.

To limit execution time and memory usage, the number of returned candidate pairs is capped via a `join_limit` parameter. This ensures the system remains responsive even when handling thousands of reviews.

Cosine similarity is applied only as a secondary filter to review pairs already selected by the LSH step. This optimization significantly reduces the number of vector comparisons required for semantic filtering.

Thanks to Apache Spark, the solution can easily be scaled to larger datasets or deployed in a distributed computing environment, making it suitable for real-world big data applications.

7 Description of the Experiments

The experiments were conducted on a subset of 1000 book reviews randomly sampled from the Amazon Books Reviews dataset. No restriction was imposed on book title or rating.

The following parameters were used during the evaluation:

- **Shingle length (`k`):** 5 characters

- **Hashing dimension (`numFeatures`):** 2^{18}
- **Number of LSH hash tables:** 5
- **Minimum Jaccard similarity threshold:** 0.2
- **Cosine similarity threshold:** 0.3
- **Maximum number of candidate pairs from LSH:** 10,000

The core similarity detection was performed using MinHashLSH, which returned candidate pairs with estimated Jaccard similarity above the threshold. These pairs were then filtered by comparing their TF-IDF vectors using cosine similarity.

Two result sets were produced:

1. Pairs selected by Jaccard similarity only.
2. Pairs that satisfied both the Jaccard and cosine similarity thresholds.

For both sets, review IDs, review texts, ratings, and similarity scores were stored. HTML exports of the results were used for qualitative inspection. The experiments were run entirely within Google Colab, with Apache Spark used in local mode. Despite hardware limitations, the system completed the analysis in a reasonable amount of time due to the use of distributed data operations and approximate similarity search methods

8 Results

id1	id2	review1	review2	rating1	rating2	jaccardSimilarity
808	871	I received the book in a timely manner and in the condition described, thank you.	The book came in great condition and in a timely manner. It was a pleasure doing business with this seller on Amazon.com	5.0	4.0	0.206452
808	896	I received the book in a timely manner and in the condition described, thank you.	book came in a timely manner and in perfect condition, the cover looks good and the pages are still crisp. Thanks!	5.0	5.0	0.271429
173	182	Like before...I am pleased to say your product was sent in a timely fashion and in very good condition. Good job...keep it up!	The book was in very good condition for an older version and it arrived in a timely manner.	5.0	5.0	0.222222
182	871	The book was in very good condition for an older version and it arrived in a timely manner.	The book came in great condition and in a timely manner. It was a pleasure doing business with this seller on Amazon.com	5.0	4.0	0.204819
182	709	The book was in very good condition for an older version and it arrived in a timely manner.	The book was delivered in good condition and in a timely fashion. I am very pleased with your services.	5.0	5.0	0.272727
173	709	Like before...I am pleased to say your product was sent in a timely fashion and in very good condition. Good job...keep it up!	The book was delivered in good condition and in a timely fashion. I am very pleased with your services.	5.0	5.0	0.300000
871	896	The book came in great condition and in a timely manner. It was a pleasure doing business with this seller on Amazon.com	book came in a timely manner and in perfect condition, the cover looks good and the pages are still crisp. Thanks!	4.0	5.0	0.203297
182	808	The book was in very good condition for an older version and it arrived in a timely manner.	I received the book in a timely manner and in the condition described, thank you.	5.0	5.0	0.242188

Figure 1: Top review pairs detected with high Jaccard similarity

id1	id2	review1	review2	jaccardSimilarity	cosine_similarity
173	709	Like before...I am pleased to say your product was sent in a timely fashion and in very good condition. Good job...keep it up!	The book was delivered in good condition and in a timely fashion. I am very pleased with your services.	0.300000	0.477406
182	808	The book was in very good condition for an older version and it arrived in a timely manner.	I received the book in a timely manner and in the condition described, thank you.	0.242188	0.519227
182	871	The book was in very good condition for an older version and it arrived in a timely manner.	The book came in great condition and in a timely manner. It was a pleasure doing business with this seller on Amazon.com	0.204819	0.410296
808	896	I received the book in a timely manner and in the condition described, thank you.	book came in a timely manner and in perfect condition, the cover looks good and the pages are still crisp. Thanks!	0.271429	0.397552
808	871	I received the book in a timely manner and in the condition described, thank you.	The book came in great condition and in a timely manner. It was a pleasure doing business with this seller on Amazon.com	0.206452	0.424542
871	896	The book came in great condition and in a timely manner. It was a pleasure doing business with this seller on Amazon.com	book came in a timely manner and in perfect condition, the cover looks good and the pages are still crisp. Thanks!	0.203297	0.390697

Figure 2: Top review pairs detected with high Jaccard similarity and Cosine similarity

9 Comments and Discussion on the Experimental Results

The MinHashLSH approach effectively identified review pairs with high syntactic similarity, particularly those sharing common phrases about delivery and product satisfaction. Applying a

cosine similarity filter further refined these results, isolating semantically coherent pairs. However, this additional step reduced the number of matches, as some lexically similar reviews used different vocabularies.

Given the project's emphasis on scalability and interpretability, Jaccard similarity was prioritized as the primary metric. Cosine similarity served as an optional refinement, applied only to Jaccard-selected pairs. This strategy balanced computational efficiency with result quality, demonstrating the pipeline's effectiveness in detecting similar reviews within a distributed processing framework.

References

[1] A. Rajaraman and J. D. Ullman, *Mining of Massive Datasets*, Cambridge University Press, 2011.

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, save and to the extent that such work has been cited and acknowledged within the text of my/our 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/us or any other person for assessment on this or any other course of study.