UNIVERSITÀ DEGLI STUDI DI MILANO

Project 1: Finding similar items

11/07/2025

AUTHORS

Zanone - 33927A

Contents

1 Introduction

In this project we tackle the problem of identifying similar book reviews within the Amazon Books Review dataset, a massive collection of user reviews related to books available on Amazon. This task falls within the broader area of text similarity detection.

The goal of this project is to implement a similarity detector capable of identifying pairs of book reviews that express similar content. The analysis is based on the Books-rating file, which contains a large collection of user reviews. To compare reviews, we represent each text as a set of tokens and compute their similarity using the Jaccard index. Since an exact pairwise comparison over the entire dataset would be computationally infeasible, we implement approximate techniques such as MinHashing and Locality-Sensitive Hashing (LSH) to efficiently retrieve candidate pairs with high similarity.

The following sections describe the methodology adopted, the preprocessing steps applied to the data, the implementation of similarity detection algorithms, and a thorough evaluation of the results obtained. Particular attention is given to ensuring that the system remains scalable and computationally efficient when applied to large datasets. This reflects the goals of the "Algorithms for Massive Data" course, which focuses on designing and implementing algorithms capable of handling and processing real-world data at scale.

2 The Dataset Processing

The analysis is based on the Amazon Books Review dataset, which initially contains approximately 3 million entries. To ensure data consistency and improve the quality of similarity detection, several preprocessing steps were applied.

First, only the relevant columns were retained: Id, Title, User_id, and review/text. All rows with missing review texts were removed. A preliminary check revealed no inconsistencies in the mapping between Id and Title, meaning each book identifier corresponded uniquely to a title.

Next, fully identical rows — i.e., entries with the same user, book, and review — were removed. A more substantial reduction came from eliminating reviews that shared the exact same review/text content across different users and books. This step alone accounted for the removal of approximately 900,000 duplicate reviews, highlighting the importance of this filter in reducing noise.

Although extremely short or trivial reviews are not removed during the initial preprocessing, the pipeline later applies a filter to retain only those reviews containing more than three tokens. This threshold is specifically chosen to ensure that at least one shingle can be generated using a sliding window of size three, which is required for the subsequent similarity computation. As a result, the analysis focuses only on reviews with a minimal amount of meaningful content while maintaining consistency with the shingling process.

After applying all cleaning operations, the dataset was reduced to roughly 2.06 million reviews, representing 68.75% of the original data.

After this sequence of filtering operations, the dataset is still relatively large. To optimize computational efficiency during the similarity detection phase, a sampling step is introduced. If enabled, a random 5% of the cleaned dataset was used, using a fixed seed to ensure reproducibility. However, the pipeline is designed to scale seamlessly to the full dataset by simply disabling the sampling flag, ensuring that all stages remain applicable to large-scale textual data.

3 Theoretical Foundations and Methodology

To identify similar reviews at scale, the project combines techniques from natural language processing and approximate similarity detection. The pipeline is designed to process raw review text and detect pairs of reviews with high textual similarity, leveraging efficient algorithms such as MinHash and Locality-Sensitive Hashing (LSH). This section describes the theoretical concepts behind each main component of the implemented pipeline.

3.1 Text Preprocessing

The raw text from each review is first processed using a sequence of natural language processing steps to normalize and clean the data.

Document Assembly

Each review is first transformed into a Spark NLP document, making it compatible with the processing pipeline.

Tokenization

The review text is split into individual tokens (words). This step is essential to break down unstructured text into manageable components.

Normalization

Tokens are converted to lowercase and stripped of punctuation. This ensures that comparisons are case-insensitive and not biased by formatting.

Stopword Removal

Common stopwords such as “the”, “and”, “is”, etc., are removed from the tokenized reviews. These words carry little semantic content and may add noise when comparing texts.

Finishing

A Finisher component extracts clean string tokens from the annotated columns and prepares them for downstream processing.

3.2 Shingling

After all the previous steps, each review is represented by a set of *shingles*, which, substantially, are contiguous sequences of $k = 3$ tokens. For instance, the sentence “I really enjoyed this book”

becomes: “I really enjoyed”, “really enjoyed this”, “enjoyed this book”. Shingling helps capture partial phrase structure and improves robustness to minor wording changes.

3.3 Feature Vectorization with HashingTF

The set of shingles for each review is converted into a numerical vector using the `HashingTF` method. This approach maps each shingle to a fixed-size feature vector using a hash function, reducing dimensionality and enabling scalable computation.

3.4 MinHash and Locality-Sensitive Hashing (LSH)

To efficiently detect similar reviews, we use MinHashLSH, a technique that approximates the Jaccard similarity between sets in sub-quadratic time.

- **Jaccard Similarity:** Given two sets A and B, the Jaccard similarity is defined as:

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

  This measure is well-suited for comparing sets of shingles.

- **MinHash:** MinHashing compresses each shingle set into a fixed-size signature such that the probability of two sets sharing the same signature component is equal to their Jaccard similarity.
- **LSH:** Locality-Sensitive Hashing groups reviews with similar MinHash signatures into the same buckets, drastically reducing the number of comparisons by focusing only on likely candidate pairs.

3.5 Similarity Detection

The MinHashLSH model is applied to the transformed dataset to compute approximate similarities between all pairs of reviews. A similarity threshold is used to filter only those pairs whose estimated Jaccard distance is below a predefined cutoff (e.g., 0.2). To avoid self-comparisons and duplicates, only pairs with distinct IDs are taken into consideration.

4 Results and Analysis

After completing the pipeline and applying the MinHashLSH model, we obtained a set of candidate pairs of similar reviews based on an approximate Jaccard similarity threshold of 0.2. To ensure that the results did not contain trivial duplicates or symmetric pairs, we filtered out comparisons involving the same review (i.e., identical `Id`) and retained only pairs where the first review's `Id` was strictly smaller than the second's.

The final output includes a selection of review pairs along with a preview of each review (up to 200 characters) and their estimated Jaccard distance. This allows for a qualitative inspection of the retrieved results to assess their lexical or semantic similarity.

To improve efficiency and reduce computation time during development, the analysis was conducted on a 5% sample of the cleaned dataset. Nonetheless, the pipeline is designed to scale seamlessly to the full dataset by simply disabling the sampling flag, ensuring that all stages can handle massive volumes of textual data when required.

A sample of the top results revealed meaningful lexical overlap, particularly in reviews expressing similar sentiments or using common phrasing. Although Jaccard-based methods (even when approximated via LSH) are not designed to capture deep semantic meaning, they offer a computationally efficient way to detect redundant or near-duplicate content.

On average, the retrieved review pairs had a Jaccard distance of approximately 0.1615, reflecting a moderate degree of lexical overlap across the dataset.

4.1 Token Length Difference Analysis

To further investigate the nature of the retrieved review pairs, we computed the absolute difference in the number of tokens between the two reviews in each matched pair. This metric, denoted as `token_diff`, helps assess whether highly similar reviews (in terms of Jaccard distance) tend to also be similar in length.

Using this additional feature, we performed both descriptive and correlation analysis. The average `token_diff` across all matched pairs was approximately 808 tokens, with a standard deviation of around 646, indicating significant variability in review lengths. The minimum difference was 0 (i.e., reviews of identical length), while the maximum reached nearly 2000 tokens.

We also computed the correlation coefficient between `JaccardDistance` and `token_diff`, obtaining a value of 0.0193. This near-zero correlation suggests that the similarity score is largely independent of the difference in review length. In other words, two reviews can be highly similar (low Jaccard distance) even if their lengths differ substantially.

This is further supported by individual examples in the data. Among the top-matching pairs (Jaccard distance close to 0), we found both short and long reviews with nearly identical token counts,

as well as some pairs where one review was significantly longer than the other. For instance, one pair with a Jaccard distance below 0.1 had a token difference of over 1300 tokens, showing that lexical overlap can occur even when one review is far more detailed than the other.

These findings highlight the robustness of the LSH-based similarity detection to differences in text length, while also offering a useful diagnostic feature for further filtering or ranking results.

5 Conclusion

This project presented a scalable and efficient pipeline for detecting similar or near-duplicate textual reviews using MinHash Locality-Sensitive Hashing (LSH) on a real-world Amazon Books dataset. Through a series of preprocessing steps—including deduplication, token filtering, and review length constraints—the dataset was cleaned and prepared for similarity analysis based on Jaccard distance over 3-word shingles.

To optimize development time and resource usage, a 5% sample of the dataset was used for testing; however, the modular design of the pipeline allows for seamless scaling to the full dataset simply by toggling the sampling configuration. The application of MinHashLSH allowed us to approximate Jaccard similarity in a computationally efficient manner, retrieving meaningful review pairs with significant lexical overlap, even when the reviews differed considerably in length.

Additional analysis on token length differences demonstrated that Jaccard similarity is largely independent of review length, with many highly similar pairs featuring substantial differences in verbosity. This confirms the robustness of the approach in detecting content-level redundancy rather than surface-level structural similarity.

Overall, this approach offers a practical solution for identifying similar user-generated content at scale, which could be valuable in applications such as content moderation, recommendation systems, or opinion summarization. Future work may explore integrating semantic embeddings to enhance detection of paraphrased but semantically equivalent reviews.