Finding Similar Book Reviews Using MinHash and Locality Sensitive Hashing

Hilal Mente
Master in Data Science for Economics

Academic Year 2024/2025

1 Introduction

The aim of this project is to identify pairs of similar book reviews from the Amazon Books Review dataset. Due to the scale of the dataset, traditional pairwise comparison (brute force) approaches are computationally infeasible. I therefore implement a scalable method combining MinHash and Locality Sensitive Hashing (LSH) to approximate Jaccard similarity efficiently.

2 Dataset Description

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 full dataset contains approximately 3 million rows.

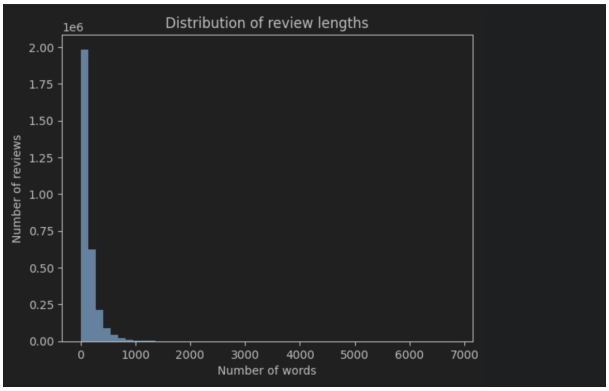

Figure 1: Distribution of review lengths in terms of number of words. Most reviews are very short, peaking below 200 words.

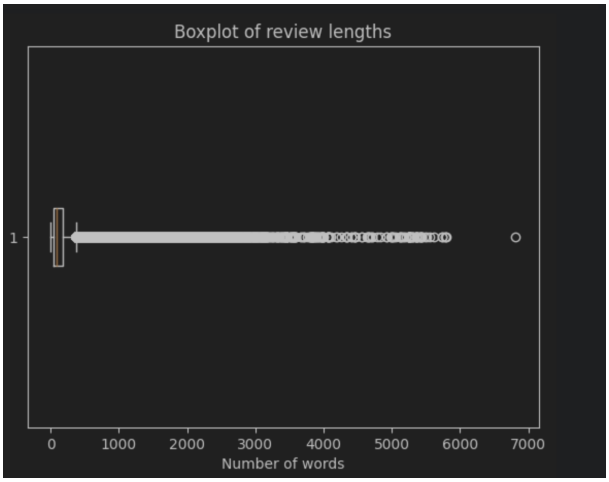

Figure 2: Boxplot of review lengths, highlighting the presence of outliers.

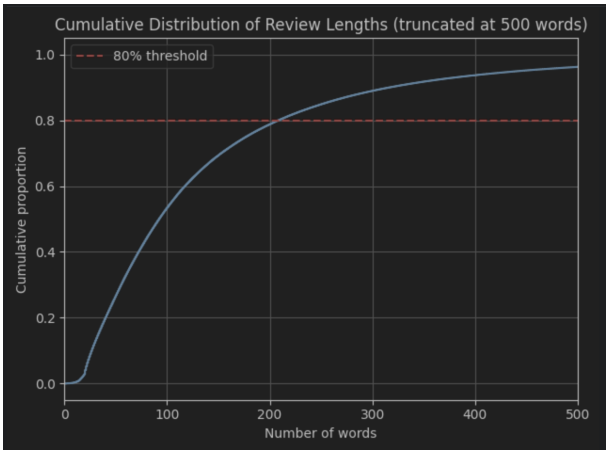

Figure 3: Cumulative distribution of review lengths (truncated at 500 words). Approximately 80% of reviews are shorter than 200 words.

3 Data Sampling

A `sample_ratio` parameter is introduced to scale down the dataset for experimentation purposes. To ensure reproducibility across different runs, I fix the random seed using `random_state=42`.

By setting a random seed, I ensure that the same subset of reviews is selected each time, making the experiments consistent and comparable.

A sample ratio parameter is introduced to scale down the dataset for experimentation:

```
sample = df.sample(frac=sample_ratio, random_state=42)
```

4 Data Preprocessing

Text Preprocessing Pipeline

Before computing similarity between book reviews, I applied a text preprocessing pipeline to clean and normalize the data. The pipeline includes the following steps:

1. **Artifact Decoding:** HTML entities (e.g., `&`) are decoded, and Unicode artifacts are normalized to improve readability and consistency.
2. **Cleaning:** All reviews are lowercased, and non-alphabetic characters (such as punctuation and numbers) are removed. Multiple whitespace characters are replaced by a single space.
3. **Filtering:**
   - Reviews that are empty or missing are discarded.
   - Reviews shorter than `k = 5` characters are removed, as they are too short to generate meaningful shingles of length 5.
4. **Deduplication:** Reviews with identical cleaned content are dropped to avoid unnecessary duplication in the similarity computations.

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

5 Similarity Detection using MinHash + LSH

MinHash is used to create compact signatures for each cleaned review using shingling (character-based), and LSH is used to efficiently query for approximate matches:

- Shingle size: 5-character substrings
- Number of hash permutations: 64
- LSH similarity threshold: 0.5

Shingling: What It Is and Why It Matters

Shingling is a technique that represents a text as a set of overlapping substrings of fixed length, typically known as *character-based n-grams*. In this project, I used 5-character shingles. For example, the word ‘‘`excellent`’’ would be transformed into the following 5-character shingles:

{`excel, xcell, celle, ellen, llent`}

This representation helps capture similarity between texts even when they contain minor variations, such as typos, plural forms, or punctuation differences.
In this project, I applied **character-level shingling with** $k = 5$ on the cleaned reviews before computing Jaccard similarity. This approach enabled the detection of similar reviews even when surface forms varied slightly due to different wording or formatting.

Why Threshold

The threshold parameter in LSH determines how similar two documents must be (in terms of Jaccard similarity) to be considered as a match. I set this value to 0.5 based on the following considerations:

- It allows for detecting reviews with moderate to high lexical overlap, even if they are not identical.
- It prevents the detection of noisy matches which might occur at lower thresholds.
- It is a common default in the literature for tasks involving near-duplicate detection.

This threshold strikes a balance between precision and recall for our lexical similarity detection task.

The function returns matched review pairs and the time taken to process the current sample size.

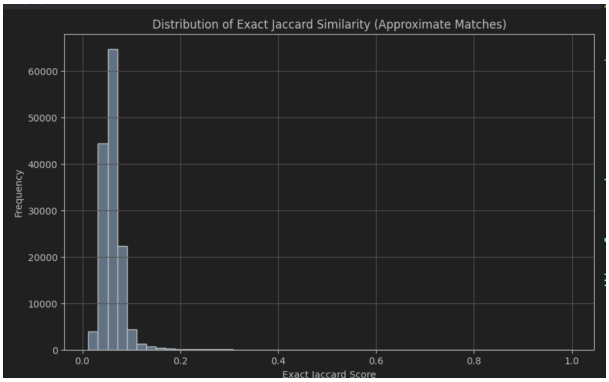

Figure 4: Distribution of actual Jaccard similarities for pairs matched using MinHash-LSH.

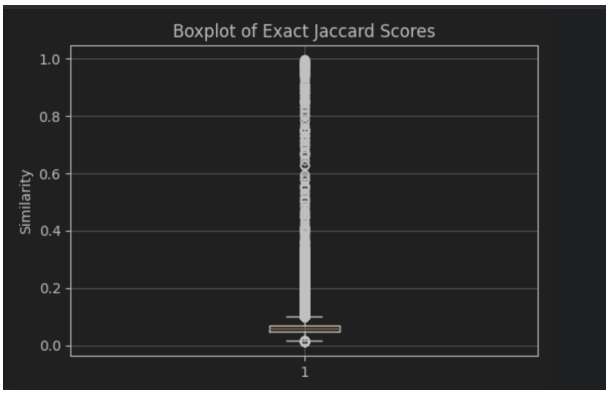

Figure 5: Boxplot of actual Jaccard similarity scores for matched pairs.

6 Case Study: Manual Inspection of Similar Reviews

To better understand the effectiveness and limitations of our similarity detection approach, I manually inspected several review pairs with high Jaccard similarity scores. The table below shows five examples, including their exact Jaccard similarity scores and short excerpts for clarity.

doc_i	doc_j	Jaccard	Excerpt from Review_i	Excerpt from Review_j
10926	50717	0.996	...Isaac Asimov tells the thrilling tells the thrilling tale of human survival in an environment that is against it.	...Isaac Asimov tells the thrilling tale of human survival in an environment that is against it.
34654	53509	0.995	...This book is a "MUST HAVE" for all female clergy!	...This is a "MUST HAVE" for all female clergy!
27002	61927	0.993	...In fact, this is one artist of a few (if not the only one) who are officially recognized by Tolkien's estate as a bard of Middle Earth.	...In fact, his is one artist of a few (if not the only one) who are officially recognized by Tolkien's estate as a bard of Middle Earth.
35922	106117	0.75	arrived fast and in great shape. I will not write long reviews. You did a great job..... for my daughter	arrived fast and in great shape. I will not write long reviews. You did a great job..... dblah blah blah
30740	37422	0.81	Bad quality and wrong definitions. Authors obviously lacking elementary knowledge. Not worth wasting time and money.	Does not meet expectations. Bad quality and wrong definitions. Authors obviously lacking elementary knowledge. Not worth wasting time and money.

Table 1: Example of similar review pairs according to exact Jaccard similarity, manually inspected.

7 Scalability Experiments

To demonstrate scalability, I ran the detection pipeline over different sample ratios. The table below shows how time and number of detected similar pairs change with increasing dataset size:

Sample Ratio	Rows	Similar Pairs Found	Time (sec)
0.0075	17,532	24,578	38.14
0.0100	23,261	35,481	49.72

8 Discussion and Analysis of Results

8.1 Effectiveness of LSH-based Similarity Detection

The system successfully identified highly similar review pairs without the need for exhaustive pairwise comparisons. Experimental results showed that even for relatively small sample sizes (e.g., 0.75% of the data), thousands of similar review pairs were identified within seconds.

Moreover, exact Jaccard scores computed for a subset of the LSH-matched pairs confirmed the high quality of the approximate matches. In some cases, the similarity exceeded 0.99, showing that MinHash with Locality-Sensitive Hashing was very effective in this domain.

8.2 Precision of Approximate Matches

To further validate the system's effectiveness, I analyzed a set of top-ranked review pairs identified via LSH and computed their exact Jaccard similarity. The results revealed that many

of these approximate matches had extremely high similarity scores. This indicates that the system is particularly successful at detecting lexically similar reviews. However, as discussed in the Limitations section, the system may still miss meaningful rephrasings with low lexical overlap—highlighting the importance of integrating semantic methods in future iterations.

8.3 Limitations

- The system may miss semantically similar reviews that have low lexical overlap, such as rephrasings with different vocabulary.
- Reviews with fewer than $k = 5$ characters cannot form valid shingles, and are therefore excluded from the detection process.

8.4 Future Work: Capturing Semantic Similarity

The current system uses lexical similarity through MinHash signatures over word shingles, which performs well in detecting near-duplicates. However, it fails to detect semantically similar reviews that use different vocabulary.

To overcome this limitation, future versions of the system could incorporate embedding-based approaches, such as Sentence-BERT or Universal Sentence Encoder. These models capture the meaning of sentences in dense vector space, allowing cosine similarity to reveal semantic overlap even in the absence of lexical matches.

9 Conclusion

The goal of the project was to detect pairs of similar book reviews from a large dataset using scalable algorithms. Instead of computing exact Jaccard similarity across all possible review pairs - which would be computationally infeasible - unlike duplicate detection, which only captures exact textual copies, our method aims to identify lexically similar reviews. This is particularly valuable in large-scale datasets where exact matches are rare but similarity still matters.

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.