Similarity Evaluation of Book Reviews

Melany Yohana Gomez Herrada

July 11, 2025

1. Introduction

This project aims to analyze textual reviews of books from Amazon and determining the degree of similarity between reviews, adopting Jaccard similarity applied to tokenized versions of the review texts.

2. Dataset Overview and Preprocessing

The dataset used in this project, *Amazon Book Reviews*, consists of user-written book reviews, along with numerical ratings on a scale from 1 to 5. After downloading and extracting the dataset, it was loaded using `pandas` in Python and inspected to understand its structure.

Reviews were classified into two sentiment categories based on their scores: those with scores above 3 were labeled as **positive**, while those with scores of 3 or below were labeled as **negative**. This approach allows for first clear partitioning of the text data.

Before computing any similarity analysis, reviews were normalized through these steps :

- Conversion to lowercase
- Punctuation stripping
- Remuval of a first set of "Stopwords" (e.g., "the", "is", "in")

These steps reduce noise and ensure that the similarity measures focus on meaningful vocabulary.

3. Jaccard Similarity

To evaluate the linguistic similarity between reviews, the **Jaccard similarity coefficient** was applied. This metric measures the similarity between two sets based on the proportion of shared elements. For sets A and B, representing the tokenized words in two reviews, the Jaccard index is defined as:

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

Jaccard similarity is particularly effective for comparing categorical or binary data. The similarity between two reviews corresponds to the ratio of shared words to the total number of distinct words between them.

In this project is used to compare the subsets of positive and negative reviews, revealing overlap in vocabulary and highlighting reviews that are thematically or descriptively similar.

3.1. Pairwise Jaccard Similarity Analysis

A first experiment with Jaccard similarity was carried out using a small subset of five positive reviews. To evaluate the similarity between these reviews, all possible pairs were generated using the combinatorial function `itertools.combinations`. For each review pair, the `jaccard_similarity` function was applied to compute the similarity score based on the proportion of shared tokens over the union of tokens between the two texts. In addition to the similarity score, the common words between the two reviews were also extracted. The results were stored in a structured `pandas` DataFrame named `positive_similarity_df`. This output provided an informative summary of pairwise similarities within the subset, highlighting which reviews share the most overlapping vocabulary.

The same procedure was then replicated for a subset composed of five negative reviews. Despite analyzing only five reviews, the initial similarity scores were quite promising and provided useful insights. However, as the number of reviews increases, the results may be affected by additional vocabulary and semantic variance introduced by a larger numer of words. The next step of the analysis aimed to filter out noisy, low-information words that might inflate similarity scores without providing meaningful insight.

3.2. Similarity Analysis Using Common Word Filtering and Bigrams

The pair of reviews with the highest Jaccard similarity was ex- tracted, and their shared vocabulary was printed explicitly. This helped illuminate which specific words were most influential in driving the similarity score between review pairs and whether they held some relevance or were merely common filler words.

To improve the similarity analysis further, the top 20 most frequent words were computed separately for the positive and negative subsets. This frequency analysis aimed to show high–occurrence tokens that appeared across a wide variety of reviews, potentially inflating similarity measures without offering meaningful semantic insight. Examining these common words, it became evident that many of them were generic (e.g., `book`, `author`, `read`, etc.), and their inclusion in similarity computations could obscure more discriminative words. To address this, the existing stop–word list was expanded by manually adding several of the most frequently occurring terms identified in the previous step. This updated `custom_stopwords` list was then used in all subsequent tokenisation processes to ensure that Jaccard similarity scores would better reflect overlaps between reviews.

Moreover, the tokenisation strategy was optimized to include bigrams in addition to unigrams. The purpose of incorporating bigrams was to capture adjucent word pairs, which often encode more specific meaning rather than isolated terms.

This methodology is aligned with the concept of *shingling*, a common technique in information retrieval, where documents are transformed into sets of consecutive n –grams. By capturing overlapping sequences of tokens (unigrams and bigrams), shingling improves the robustness of similarity evaluations.

3.3. Threshold Tuning and Duplicate Detection in a Larger Subset

To evaluate the scalability of the Jaccard-based similarity approach, the review subset was expanded to include 200 positive reviews. Each review was first tokenized using the refined tokenizer incorporating unigrams and bigrams, and the pairwise Jaccard similarity was computed between all possible review pairs.

A similarity threshold of 0.2 was initially adopted to define whether two reviews could be considered similar. This allowed for the identification of three review pairs exceeding this similarity level. However, upon manual inspection of the most similar pair, indexed at 132 and 134, it became evident that the two reviews were virtually identical:

> `Kurt Seligmann, Surrealist artist par excellence, admitted & unashamed bibliophile, has ravaged his occult library in a miraculous marriage giving birth to this classic historical account of Magic.`

This observation highlighted the need to differentiate between similar content and near-exact duplicates. To address this, two thresholds were introduced :

- `duplicate_threshold = 0.95`, above which reviews are considered duplicates;
- `similarity_threshold = 0.2`, the lower bound for what is still considered a meaningful degree of similarity.

While this approach improved the classification, some duplicates still passed the similarity filter. As a result, it was necessary to refine the selection by isolating an intermediate similarity range:

```
high_sim_df = similar_df[
    (similar_df['Jaccard similarity'] >= 0.2) &
    (similar_df['Jaccard similarity'] <= 0.7)
]
```

This interval was empirically found to be optimal for capturing reviews that share semantically relevant content without falling into introduced . For example, the pair at indices 160 and 163 had a similarity score of approximately 0.555 and referred to related philosophical topics and metaphysical figures, while remaining distinct in their wording.

> **Review 160:** *Dr Baker explains clearly and engagingly how one can improve one's life by changing your subconscious pattern through the spiritual technique called treatment.*
> **Review 163:** *Dr Baker was one of those great 20th century metaphysicians like Emmet Fox, Ernest Holmes & Thomas Troward, who understood the working of the mind long before psychotherapy became popular.*

The application of this interval effectively excluded the exact duplicate pairs, preserving only substantively similar but non-identical reviews.

The same procedure was implemented for the negative reviews subset. However, when applied to a subset of 200 negative reviews, no review pairs were identified within the defined similarity range ($0.2 \leq \text{similarity} \leq 0.7$).

3.4. Similarity Analysis on 1000 Reviews and Results

In the final part of the experiment, I extended the analysis to a larger subset of 1000 positive reviews in order to test the scalability and effectiveness of the Jaccard similarity–based approach. The core steps of the algorithm remained consistent:

1. Extract a subset of reviews with scores greater than 3.
2. Tokenize each review by removing punctuation and stopwords, and generating both unigrams and bigrams via the `get_tokens` function.
3. Compute pairwise Jaccard similarity between all unique review pairs.
4. Identify:
    - Pairs of reviews with high similarity (similarity ≥ 0.2),
    - Near-duplicate reviews (similarity ≥ 0.95),
    - Moderately similar reviews in the interval $[0.2, 0.7]$.
5. Remove duplicate entries to prevent redundant patterns in further analysis.
6. Print out and manually inspect the most similar and representative pairs.

The similarity computation was carried out for all review pairs. Using a similarity threshold of 0.2, the algorithm retrieved multiple pairs with meaningful overlaps, including thirteen pairs with similarity scores ranging from 1.0 down to 0.2. Remarkably, several pairs reached a similarity of exactly 1.0, indicating near-identical or repeated texts.

In contrast, lower-similarity but still semantically linked pairs were identified in the moderate similarity range $0.2 \leq \text{similarity} \leq 0.7$. This range appeared to be optimal for surfacing reviews that address similar themes or sentiments, while excluding redundant entries.

All in all, the functions and thresholds defined throughout the analysis — particularly `get_tokens` for tokenisation and `jaccard_similarity` for set-based distance computation — successfully allowed for the detection of pairs of similar reviews, This approach enabled scalable, interpretable similarity assessments across a large dataset, and highlighted the usefulness of n-gram–based representations in exploratory textual analysis.

References

[1] Jure Leskovec, Anand Rajaraman, and Jeffrey D. Ullman. *Mining of Massive Datasets*. Cambridge University Press, 3rd edition, 2020.

[2] Sam Chahine, *NLP: How to Find the Most Distinctive Words Comparing Two Different Types of Texts*.

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/us or any other person for assessment on this or any other course of study.