NLP Book Recommendation System

Meskerem Goshime
Springboard Data Science Program
Final Capstone Project
December 2022

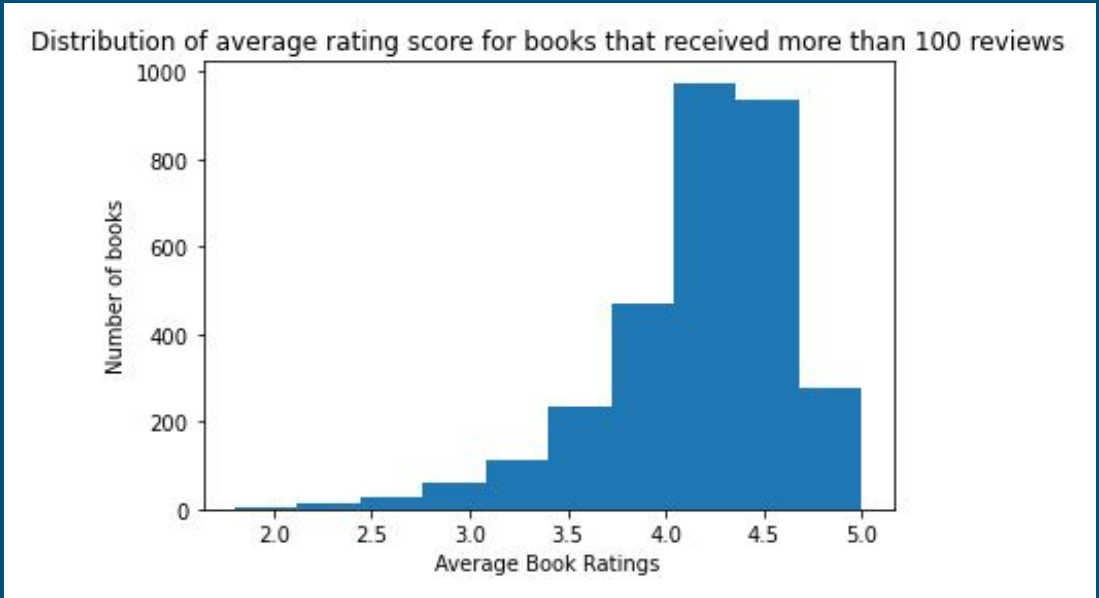
Problem Definition

- Finding the next perfect book has always been a challenge for book lovers. With the large number of books being published every year, the task of finding just the right book a particular reader will like becomes more and more difficult. If booksellers can recommend the right books for readers, they can boost their revenue tremendously.
- The purpose of this project will be to identify which books are similar based on text analysis of categories and descriptions of the books. The system will take a title of a book that the user read before and liked as an input. Then it will be able to recommend the 5 most similar books to the chosen title.

Dataset

- I used the Amazon Book Reviews data available on Kaggle here: https://www.kaggle.com/datasets/mohamedbakhet/amazon-books-reviews?select=books_data.csv
- This is a rich dataset for Natural Language Processing containing text descriptions and categories for 212,403 books as well as 3,000,000 text reviews from users. Therefore it is ideal for text analysis.
- The data originally come in two csv files, one for books data and the other for user reviews data, with the below columns.
- Books: Title, description, authors, image, previewLink, publisher, publishedDate, infoLink, categories, ratingsCount
- Ratings: Id, Title, Price, User_id, profileName, review/helpfulness, review/score, review/time, review/summary, review/text

Exploratory Data Analysis - Distribution of Average Rating for Books with more than 100 Reviews



Exploratory Data Analysis - Authors with the Highest Total Number of Reviews for All Their Books

authors	Total # of Reviews
['J. R. R. Tolkien']	37268.0
['Jane Austen']	29933.0
['Charles Dickens']	17690.0
['John Steinbeck']	15461.0
['John Ronald Reuel Tolkien']	12558.0
['Kurt Vonnegut']	12093.0
['Harper Lee']	12013.0
['C. S. Lewis']	11800.0
['F. Scott Fitzgerald']	10535.0
['George Orwell']	10182.0

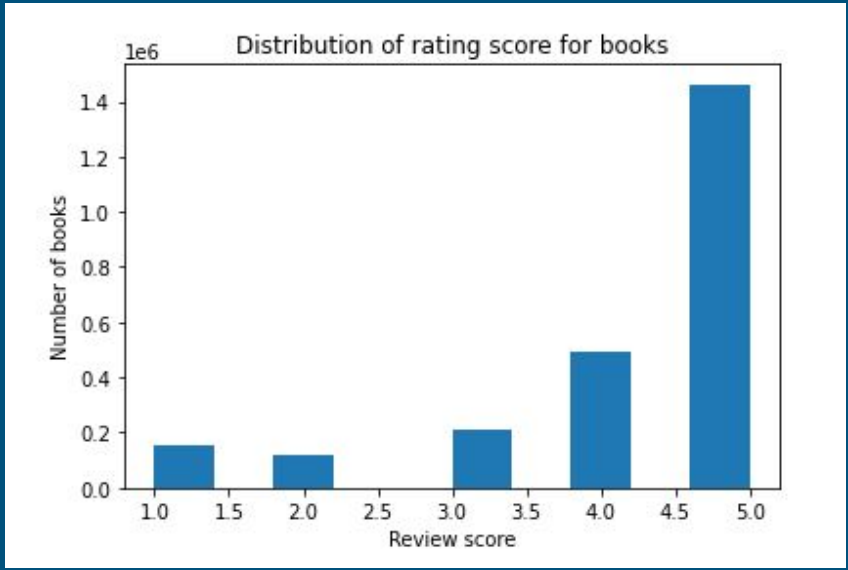
Name: review/score_Count, dtype: float64

Exploratory Data Analysis - Books with Higher Number of Reviews Received Lower Average Rating

Books which received a higher number of ratings had lower average ratings than the average rating for all the books.

	Mean Rating	**Median Rating**
Books With >10 Reviews	4.21	4.30
All Books	4.26	4.48

Exploratory Data Analysis
Ratings



- Most ratings are 4 or 5 points.
- Most users rated between 1 to 5 books. However, the highest number of ratings by one user is 5795 and the second highest is 3606. I quickly scanned these ratings and they looked genuine.

Data Cleaning - Missing Values

Title	1
review/score_Avg	1
review/score_Count	1
description	68442
authors	31413
publishedDate	25622
categories	41199
dtype: int64	

The books data had missing values as shown in the table. Here is how I handled the missing values in the various columns.

- Published date: filled the missing dates with the average published date value.
- Authors: filled missing values with ‘unknown’
- Description and categories: Since these columns are vital for text analysis, I dropped the rows with null description or categories.
- Title, review/score_avg, and review/score_count - removed the missing rows.

Additional Data Cleaning Tasks Performed

Here are the major data cleaning I performed on the data.

- I dropped the columns image, previewLink, infoLink, ratingsCount, publisher from the Books Data.
- I dropped the columns Price, profileName, review/time from the Ratings Data.
- The categories column in the books data has 5415 unique categories. However, I noticed that there are duplicate categories with slightly different wording. I did not worry about it too much because I planned to perform NLP processes on the categories column which would detect similar categories as similar to each other.
- The title column had duplicate values with slightly different wording/spelling and/or case. I converted the text in the title column to lowercase and removed duplicates. I dropped the duplicates with less number of ratings and kept the ones with the highest number of ratings. However, I did not have a good solution for the duplicate titles with slightly different wording or spellings.
- I removed empty spaces, special characters and numbers from the Title column.

Feature Engineering and Text Preprocessing

- I calculated the average review score and the total number of ratings for each book from the ratings data. Review/score_avg and review/score_count columns were then added to the Books data.
- I Converted the publishedDate column in the books data to date/time in the format of 4-digit year.
- I combined the categories and description columns and created description_categories column. This is the column on which I will do NLP processes to produce the book recommendations.
- I removed special characters and numbers from the description_categories column since these will not have value to the meaning of the text. I also converted the text to lowercase.
- I then tokenized the text in the description_categories column to separate each word in the text using word_tokenize from nltk library.
- Afterwards, I lemmatized each word using WordNetLemmatizer from the nltk library. This reduced the variant forms of a word to one form. This will help the models to recognize variant forms of words as one word and provide better performance.
- I also removed stop words (like the, and, are, is …) which do not add meaning to the text. I used the nltk.corpus to generate a list of stop words.
- In the end, I joined the words back together to form one long string for the categories_description value of each book.

Model 1 - CountVectorizer and Cosine Similarity

Here are the steps I took to make the book recommendations:

1. I used CountVectorizer to create vectors for the description_categories text.
2. I then created the Cosine Similarity matrix for all the whole data using scikit-learn's Cosine Similarity function.
3. After that, I was able to pick the row for the chosen title in the matrix using its index value. I then sort that row and take the top 5 books with the highest similarity scores with the chosen book.
4. The way this recommendation system works is the user will choose a book he/she read and liked before. The system then looks for 5 books that are most similar to that chosen title using the similarity matrix and makes recommendations.

Model 2 - Gensim and Spacy

Here are the steps I took to make book recommendations with this model:

1. I vectorized the description_categories column using nlp.
2. I then took the chosen book and computed the similarity of that book with all of the books, including itself using Spacy.
3. After that, I was able to sort the similarity scores in descending order to find the 5 most similar books to the chosen book.
4. As this model does not output a similarity matrix like in the SBERT model or the scikit-learn's Cosine Similarity, I designed the system to compute similarity for a chosen title when needed instead of calculating similarity scores of the whole data in advance. I am not worried about this because this step runs very fast.

Model 3 - with SBERT Sentence Transformer and Cosine Similarity

Like in the first two models, I took a subset of the data, which is books that received more than 10 reviews. These are the steps I took to make recommendations with this model.

1. I used SBERT SentenceTransformer pre-trained model to create sentence embeddings for the description_categories column. This created sentence embeddings for the whole data.
2. I then created the cosine similarity matrix for the whole data based on the sentence embeddings using the cos_sim function in the SBERT library.
3. After that, I was able to pick the row for the chosen title in the matrix using its index value. I then sort that row and take the top 5 books with the highest similarity scores with the chosen book.
4. The way this recommendation system works is the user will choose a book he/she read and liked before. The system then looks for 5 books that are most similar to that chosen title and make recommendations.
5. Creating the sentence embeddings and similarity matrix takes some time to run. However, once the matrix is created, making recommendations is very fast.

Choosing Model 1, Model 2 or Model 3?

I have finally decided to go with the SBERT Sentence Transformer pretrained model (Model 3) because of the following reasons.

1. According to my test recommendations, the Gensim and Spacy model (Model 2) seem to produce lower quality recommendations. Although I liked the simplicity of the model and how fast it runs, I did not choose it due to the lower quality of its recommendations.
2. Although the Count Vectorizer model (Model 1) provides impressively decent recommendations, the Sentence Transformer model (Model 3) seems to provide even better recommendations.
3. The SBERT Sentence Transformer Model (Model 2) recognizes synonyms and takes into consideration the semantic and contextual meanings of words and sentences. In addition, SBERT is specifically designed to analyze contextual meanings of sentences as opposed to focusing on just words. The Count Vectorizer (Model 1) and the Gensim and Spacy (Model 2) are very simplistic and only counts how often each word appears in each text and identifies similar text based on commonly appearing words. Therefore, it is not surprising that SBERT seems to perform better.
4. In all I am impressed with how all of the models were able to analyze text information and pick out similar books out of thousands of books.

How About Accuracy Matrix?

In the absence of labeled data, I was not able to quantify the accuracy of the models. However, I have assessed the recommendations for several books. I have presented my assessments of three example recommendations using each model. Please see the table summarizing my assessments in the next slide.

Scores given: Not good, Can be better, Good, Very good

Note: I do have experience with books as I have worked as a reference librarian. However this is still a subjective assessment and may not be free of bias. In addition, I have only assessed a very tiny fraction of the recommendations.

	1 is One	Spanish Step by Step	To Kill a Mockingbird
Description of the chosen book	Early childhood rhyming picture book.	Spanish language learning book.	In an Alabama town during the Great Depression, a white lawyer defending a black man accused of raping a yount white women.
Model 1	**Very good** All early childhood picture books, several in rhyme/riddle format.	**Good** All language learning books and 3 out of 5 for the Spanish language specifically.	**Good** Mostly older (18th and 19th Centuries) popular fictions (one biography), but I do not see a lot of overlap on their topics.
Model 2	**Not good** One of the recommendations, The dictionary of the khazars is a very different book from the chosen book. It is a 339 page novel. The rest of the recommendations are rhyming early childhood picture books	**Not good** All of the recommended books are language learning books, but none are for the Spanish language.	**Good** The recommended books are similar to the chosen title in that they have some element of crime and mystery and two of the books have some multi-etinic elements like the chosen title. However, they mostly lack the slavery theme.
Model 3	**Very Good** All rhyming early childhood picture books.	**Can be better** All language learning books, out of which 2 are for Spanish language and 3 are for other languages.	**Very good** Historical fictions that deal with the topics of slavery, race relations, everyday life in the 17th Century and early 18th Century America.

How about the book reviews data

Initially, my plan was to also work on the review texts of the reviews data. However, the books data with 138000 rows required 76 gb of memory. However, with the reviews data containing more than 2 million rows, it requires an insane amount of memory. Therefore, I am going to base my recommendations on just the categories and description columns of the books data. In addition to that, I think the categories and descriptions are better data to understand what a book is about than reviews.

Limitations

1. I have not found a good solution for duplicate titles with spelling/wording differences.
2. I have not done a quantifiable performance measure due to the absence of labeled data.
3. I wish I was able to do this project with a cleaner dataset. For example, if I would be able to do this project with a dataset from a library catalog, it would usually have subject headings for the books which is a standardized list of subjects. The title and author columns would also be entered in a standardized format which would allow easy detection and handling of duplicates.
4. This dataset contains a very rich book reviews data. However, I only used the books data with categories and description columns. I have not used the reviews data apart from computing average ratings and total number of ratings for each book which I added to the books data.
5. I have not created a search feature where the user will enter a title. This will need to consider spelling errors and variant titles. This is out of the bounds of this project. However, it will need to be done if this project is to be implemented.

References

- AtoZdatabases. (n.d.). Ultimate Reference Tool for Academic Research. AtoZdatabases Academic| The Premier Job Search, Reference and Mailing List Database. Retrieved November 30, 2022, from http://www.atozacademics.com/home
- Computing sentence embeddings¶. Computing Sentence Embeddings - Sentence-Transformers documentation. (n.d.). Retrieved November 28, 2022, from https://www.sbert.net/examples/applications/computing-embeddings/README.html
- Edumunozsala. (n.d.). Intro-nlp-text-classification/intro_nlp_1_tfidf_text_classification.ipynb at master · Edumunozsala/Intro-NLP-text-classification. GitHub. Retrieved November 28, 2022, from https://github.com/edumunozsala/Intro-NLP-Text-Classification/blob/master/Intro_NLP_1_TFIDF_Text_Classification.ipynb
- Industry market research, reports, and Statistics. IBISWorld. (2022, September 21). Retrieved November 30, 2022, from https://www.ibisworld.com/united-states/market-research-reports/book-publishing-industry/
- Jha, A. (2022, November 14). Vectorization techniques in NLP [guide]. neptune.ai. Retrieved November 29, 2022, from https://neptune.ai/blog/vectorization-techniques-in-nlp-guide#:~:text=Vectorization%20is%20jargon%20for%20a%20classic%20approach%20of,various%20domains%2C%20and%20it%E2%80%99s%20now%20used%20in%20NLP.
- Malik, U. (2022, July 21). Python for NLP: Creating bag of words model from scratch. Stack Abuse. Retrieved November 28, 2022, from https://stackabuse.com/python-for-nlp-creating-bag-of-words-model-from-scratch/
- Python: NLP analysis of Restaurant Reviews. GeeksforGeeks. (2021, November 2). Retrieved November 28, 2022, from https://www.geeksforgeeks.org/python-nlp-analysis-of-restaurant-reviews/?ref=lbp
- rashida048. (2019, December 12). Movie Recommendation Model Using Cosine_Similarity and CountVectorizer: Scikit-Learn. Regenerative. Retrieved November 28, 2022, from https://regenerativetoday.com/movie-recommendation-model-using-cosine_similarity-and-countvectorizer-scikit-learn/
- Thursday Content. (2021, March 23). Sentence similarity using Gensim & Spacy in python. YouTube. Retrieved November 29, 2022, from https://www.youtube.com/watch?v=Il04RjS-9-8&t=262s