NLP Book Recommendation System

Meskerem Goshime
Springboard Data Science Program
Final Capstone Project
December 2022

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

Data Cleaning Tasks Performed

- Dropped columns from the Books Data: image, previewLink, infoLink, ratingsCount, publisher
- Dropped Columns from the Ratings Data: Price, profileName, review/time
- 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 and text similarity process 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.

Missing Values

- The books data had the following missing values:

```
1  books.isna().sum()
```

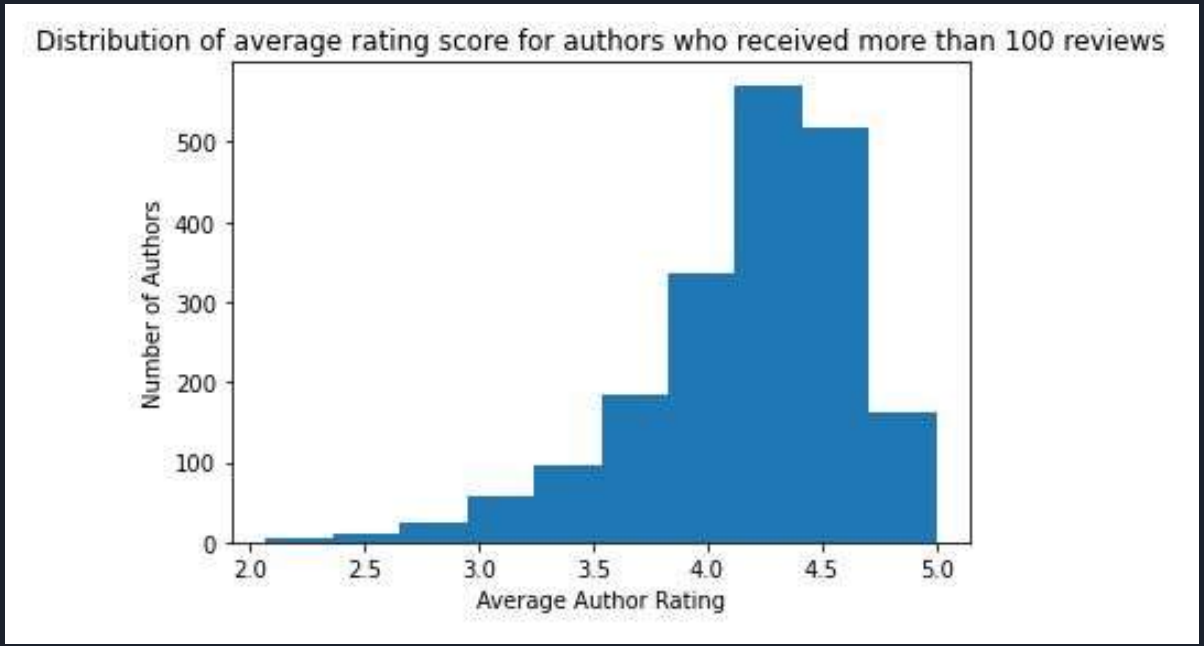
```
Title                1
review/score_Avg     1
review/score_Count   1
description      68442
authors          31413
publishedDate    25622
categories       41199
dtype: int64
```

- 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.

Feature Engineering

- I used groupby to calculate 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.
- Converted the publishedDate column in the books data to date/time in the format of 4-digit year.

EDA - Distribution of average rating for authors with more than 100 reviews.



Authors with the highest number of reviews

authors	
['Agatha Christie']	112
["Louis L'Amour"]	110
['William Shakespeare']	106
['Ann M. Martin']	65
['Carolyn Keene']	62
['Edgar Rice Burroughs']	59
['Various']	58
['DK']	55
['Nora Roberts']	52
['Mark Twain']	51

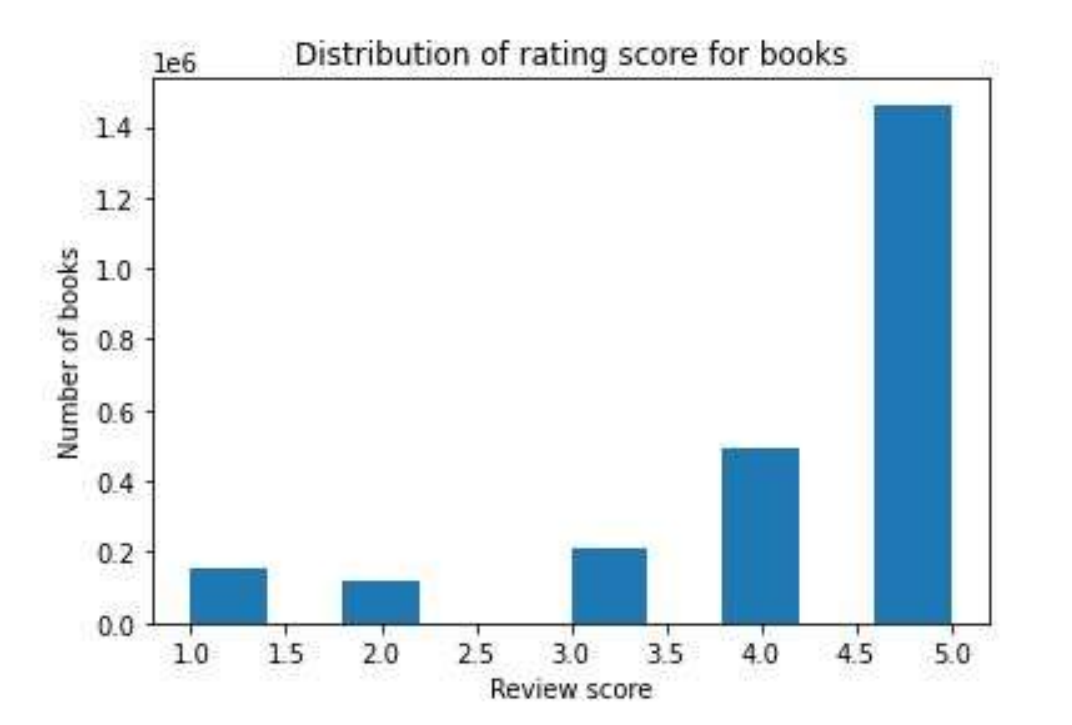
EDA - review/score count and review/score average

There is not a strong correlation between how many reviews a book received and the average review score.

	review/score_Avg	review/score_Count
review/score_Avg	1.000000	-0.008262
review/score_Count	-0.008262	1.000000

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.

Text Preprocessing

- I combined the categories and description columns and created categories_description column.
- I removed special characters and numbers since these will not have value to the meaning of the text. I also converted the text to lowercase.
- I then tokenized the text to separate each word in the text.
- I lemmatized each word in order to group various forms of a word together. This will help the model to recognize similar words as one word and provide better performance.
- I also removed stop words which do not add meaning to the text.
- 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

- Since the books data was too large to do cosine similarity on, I took a subset of the book data, which is books with more than 10 reviews. There are 29560 books in this smaller dataset.
- I also moved to a Google Colab Pro environment in order to get access to better RAM and GPU.
- I used CountVectorizer to create vectors for the description_categories text.
- I then used created the Cosine Similarity matrix.
- After that, I was able to pick the 5 books with highest similarity scores to the chosen title in order to make recommendations. 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 are are most similar to that chosen title using the similarity matrix and make recommendations.

Model 1 - How about accuracy matrix

In the absence of labeled data, I was not able to quantify the accuracy of the model. However, I have assessed the recommendations for several books and it does seem to make some decent recommendations most of the time.

Here is an assessment of three example recommendations made using this model.

1. 1 is one - the chosen book is early childhood rhyming and counting picture book. The recommended titles are all early childhood picture books, several in rhyme/riddle format which matches with the chosen book. I think the recommendations for this book are great!
2. Spanish stepbystep - In this example, the recommended books are all language learning books and many of them for the Spanish language. It would have been better if they were all for the Spanish language though.
3. To kill a mockingbird - The recommended books are all mostly older (18th and 19th Centuries) popular fictions (one biography), but I do not see a lot of overlap on their topics. Therefore, I am not completely satisfied with this recommendation.

Model 2 - with SBERT Sentence Transformer and Cosine Similarity

- Like in the first model, I took a subset of the data, which is books that received more than 10 reviews. There are 29560 books in the list.
- I then used SentenceTransformer pre trained model to create sentence embeddings for the description_categories column.
- I then created the cosine similarity matrix based on the sentence embeddings.
- After that, I was able to pick the 5 books with highest similarity scores to the chosen title in order to make recommendations. 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 are are most similar to that chosen title and make recommendations.

Model 2: How about accuracy matrix

In the absence of labeled data, I was not able to quantify the accuracy of the model. However, I have assessed the recommendations for the same 3 books I used as an example for Model #1 so that I can compare the two models.

1. 1 is one - the recommendations for this book are all rhyming early childhood picture books. I think this is an excellent recommendation. This particular recommendation is comparable to the 1st Model.
2. Spanish stepbystep - the recommendations for this book are all language learning books, out of which 2 are for Spanish language and 3 are for other languages. The result seems similar to the 1st Model.
3. To Kill a Mockingbird - the recommendations for this book are historical fictions that deal with the topics of slavery, race relations, everyday life in the 17th Century and early 18th Century America. I think it is a pretty good recommendation list and better than the 1st Model.

Choosing between Model 1 and Model 2

I have finally decided to to with the SBERT Sentence Transformer pretrained model due to the following reasons.

- As assessed manually, although the Count Vectorizer model (Model 1) provides impressively decent recommendations, the Sentence Transformer model (Model 2) seems to provide even better recommendations.
- The Sentence Transformer Model (Model 2) takes into consideration the meanings of words how much different words are close to each other. For example, it considers synonyms. However, the Count Vectorizer (Model 1) only counts how often each word appears in each text and identifies similar text based on commonly appearing words.

How about the book reviews data

Initially, my plan was to also to 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. 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.

References

- 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
- 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/