Meskerem Goshime
Springboard Data Science Program
Final Capstone
December 2022

NLP-Based Book Recommendation System

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 becomes more and more difficult. If libraries or book sellers can recommend the right books for their readers, they can boost their readership and revenue tremendously.

According to a report on IBIS website, Market size of the U.S. book publishing industry is $30 billion. Amazon Publishing alone has an annual revenue of $5,250,000,000 according to AtoZ Databases. Currently, most readers rely on recommendations of librarians or websites like goodreads and LibraryThing Book Suggester. It is important for book sellers and libraries to have a good book recommendation system to help users find the right books and thereby increase their book sales or readership.

For this project, I used Amazon Books Reviews data by Mohamed Bakhet. I chose this dataset because it includes categories and the text description for 212,404 books and 3,000,000 ratings and text reviews from users. Therefore, it is ideal for text analysis using Natural Language Processing tools, which I am interested in.

The purpose of this project is to identify which books are similar by analyzing text descriptions of the books using Natural Language Processing (NLP) machine learning algorithms. The system will take a title of a book that the user read before and liked as an input. Then it will recommend the 5 most similar books to the chosen title.

Why Natural Language Processing (NLP)?

Recommending the right book is a difficult task because books and readers are both complex and unique. NLP makes it possible for a machine to understand human language, even understand context, and determine similarity between blocks of texts. This is a content based

recommendation approach as opposed to a collaborative recommendation approach. Hence I am saying “you liked this item before, so you will probably like this other item which is similar to the one you liked.” Content based recommendation approach can be precise as long as we can effectively determine how similar items are.

Data about books like number of pages, publisher, publication date, audience etc is inadequate to describe books. Books are text data! Looking at the content of the book is the best way to know what the book is about. If we cannot look at the whole book, the next best thing we can do is look at summaries, descriptions and categories of books.

Analyzing text data is a complex task in itself. However, NLP makes it possible for us to analyze text data and understand human language. With my approach, the user has to provide a book title that he/she liked before. However, this information may also come from the user’s previous ratings data on books.

Collaborative approach may also work for book recommendation and may also be used in conjunction with such NLP content based recommendation system.

For this project, I used Python and the various Python data science libraries to clean, analyze, and prepare the data for machine learning models. I then built AI models using Count Vectorizer, Spacy and SBERT algorithms to determine similarities of books and make recommendations.

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

1. 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
```

The books data had the above shown missing values. 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.

2. Additional Data Cleaning Tasks Performed

Here are the major data cleaning tasks 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.

EXPLORATORY DATA ANALYSIS

1. Books With The Highest Number Of Ratings

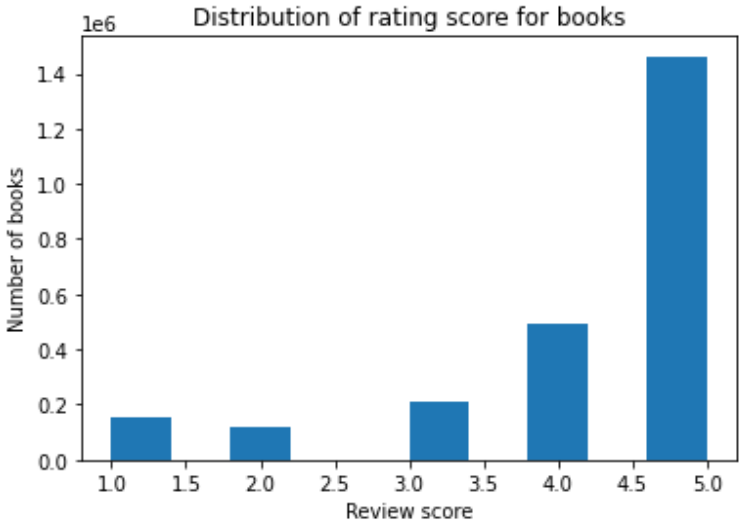
	Title	authors	review/score_Count	review/score_Avg
0	The Hobbit	['J. R. R. Tolkien']	22023.0	4.657131
1	Pride and Prejudice	['Jane Austen']	20371.0	4.527662
2	The Giver	['Lois Lowry']	7644.0	4.273417
3	Great Expectations	['Charles Dickens']	7421.0	4.089880
4	Harry Potter and The Sorcerer's Stone	['J. K. Rowling']	6796.0	4.739258
5	Brave New World	['Aldous Huxley']	6312.0	4.235266
6	Mere Christianity	['C. S. Lewis']	6053.0	4.469684
7	The Great Gatsby	['F. Scott Fitzgerald']	5291.0	4.151200
8	To kill a mockingbird	['Harper Lee']	4805.0	4.546514
9	The Hobbit There and Back Again	['John Ronald Reuel Tolkien']	4438.0	4.657503

2. Authors with the Highest Total Number of Reviews for All Their Books

Authors of popular older classics seem to have received the highest number of reviews.

```
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
```

3. 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.

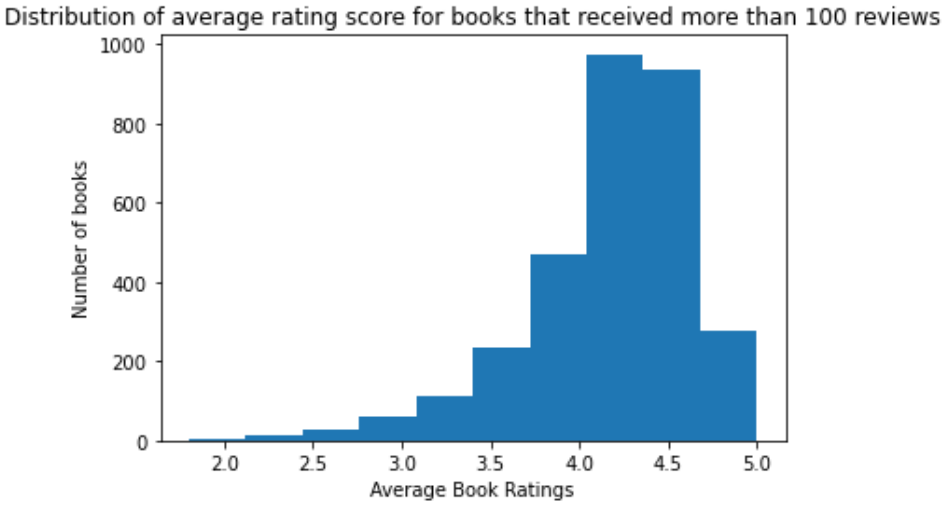
4. Books with Higher Number of Reviews Received Lower Average Rating

Books which received a higher number of ratings had lower average ratings than the average rating for all the books. This makes sense because books with 1 or 2 ratings are likely to receive all 5 point ratings, which skews the data. When a book has a large number of ratings, it is likely to receive a few of the lower ratings.

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

5. Distribution of Average Rating for Books with More Than 100 Reviews

Most books received an average review of roughly between 3.75 and 4.75.



FEATURE ENGINEERING AND TEXT PREPROCESSING

Here are the major feature engineering and text preprocessing tasks performed.

- 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.

MAKING THE RECOMMENDATIONS

I built three models that make the book recommendations and compared their performance. The models take a book title from the user that they read and liked before as an input. The system then looks for five books that are most similar to that chosen title and make recommendations.

Since the books data was too large to process, 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 have access to better RAM and GPU.

MODEL 1 - COUNT VECTORIZER AND COSINE SIMILARITY

Here are the steps I took:

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 took the row for the chosen title in the matrix. From that row, the 5 highest similarity score values correspond with the books that are most similar to the chosen title.

MODEL 2 - SPACY

Here are the steps I took to make book recommendations:

1. I vectorized the description_categories column using nlp.
2. I then computed the similarity score of the chosen 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

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 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. This model runs faster as compared to Model 1.

HOW ABOUT ACCURACY MATRIX?

I used Discounted Cumulative Gains (DCG) to measure and compare performances of the three models. I manually went through the description of the chosen titles and compared them to the descriptions of the recommended titles to determine relevancy score.

DCG (Discounted Cumulative Gains) = Relevancy Score/(LOG(Recommendation Rank+1))

Formula Source: Towards Data Science, [An Exhaustive List of Methods to Evaluate Recommender Systems]

Scoring Key:

Most relevant 2 Somewhat relevant 1 Least relevant 0

		Count Vectorizer	Count Vectorizer	Spacy	Spacy	SBERT	SBERT
Chosen Book Title	**Recomm-endation Rank**	**Relevancy Score (CG)**	**DCG**	**Relevancy Score (CG)**	**DCG**	**Relevancy Score (CG)**	**DCG**
1 is one	1	2	6.64	2	6.64	2	6.64
	2	2	4.19	2	4.19	2	4.19
	3	2	3.32	0	0.00	2	3.32
	4	2	2.86	0	0.00	2	2.86
	5	2	2.57	2	2.57	2	2.57
spanish stepbystep	1	2	6.64	1	3.32	2	6.64
	2	1	2.10	1	2.10	2	4.19
	3	2	3.32	1	1.66	1	1.66
	4	2	2.86	1	1.43	1	1.43
	5	1	1.29	1	1.29	1	1.29
to kill a mockingbird	1	2	6.64	2	6.64	2	6.64
	2	1	2.10	2	4.19	2	4.19
	3	1	1.66	2	3.32	2	3.32
	4	1	1.43	1	1.43	2	2.86
	5	0	0.00	2	2.57	2	2.57
Total		**23**	**47.63**	**20**	**41.36**	**27**	**54.39**

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 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 CountVectorizer 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 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.
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 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 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.

3. This dataset contains a very rich book ratings/reviews data. I only extracted average ratings and total number of ratings from the ratings data, but largely used the books data. Other NLP projects can be done using the ratings/reviews data.
4. I have not created a data input interface for users. This is out of the bounds of this project. But, if implemented, the system will need to consider spelling errors and variant titles.

REFERENCES

- AtoZdatabases. (n.d.). *Ultimate Reference Tool for Academic Research*. AtoZdatabases Academicl 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=ll04RjS-9-8&t=262s

ACKNOWLEDGEMENTS

Huge thanks:

- To Springboard for taking me so far into the data science field.
- To my Springboard mentor, Tony Paek, for his constant guidance and encouragement.
- To Mohamed Bekheet for making the Amazon Books Reviews dataset available on Kaggle.
- To the many data scientists who wrote articles, shared their codes and recorded video lessons on NLP. I learned from all of you!
- To my wonderful husband, Sami, for always putting me before himself!
- To my sons Peter and Caleb for encouraging me and for being there for me.
- To my brother Yonatan Goshime for putting the idea of getting into the data science field in my head.