# NLP-Based
# Book Recommendation System

Meski 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.
- If libraries or book sellers can recommend the right books for readers, they can boost their readership and revenue tremendously.
- The purpose of this project is to identify which books are similar by analyzing natural language text descriptions of the books using 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 NLP?

- Recommending the right book is a difficult task because readers and books are both complex and unique.
- Content based approach can be precise if we can determine how similar the books are.
- Books are text data, and text descriptions are best in describing books.
- NLP makes it possible for a machine to understand human language, even understand context, and determine similarity between blocks of texts.
- If the models built in this project get implemented, previously liked book can come as a user input or can come from user ratings.

# 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
- Rich dataset for Natural Language Processing containing text descriptions and categories for 212,403 books as well as 3,000,000 text reviews.
- The data come in two csv files, books data and user reviews data.
- 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 - Missing Values

```
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.

# Additional Data Cleaning Tasks Performed

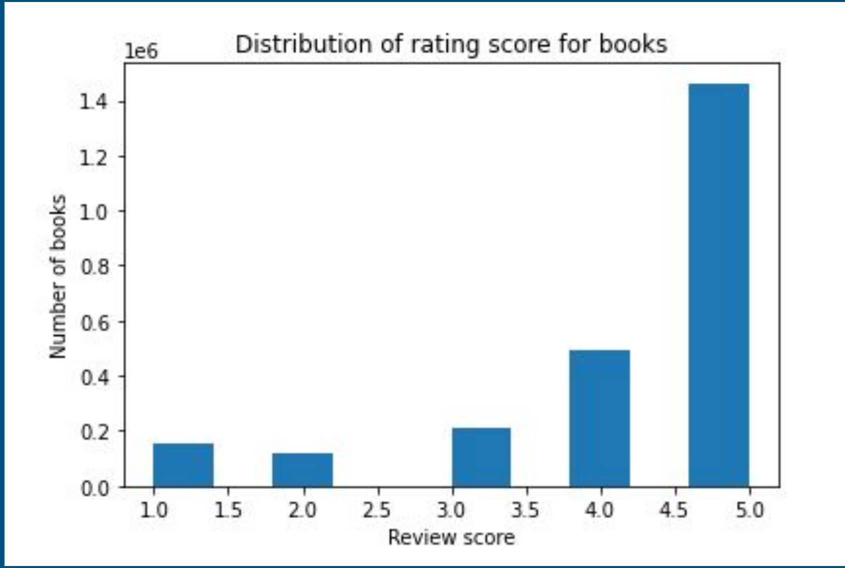
Major data cleaning tasks performed.

- Dropped unneeded columns.
- Categories column
  - has 5415 unique categories.
  - has duplicate categories with slightly different wording.
- Title column - some duplicate titles with slightly different wording/spelling and/or case.
  - Converted to lowercase and removed duplicates.
  - Still some duplicate titles with slightly different wording/spellings.
- I removed empty spaces, special characters and numbers from the Title column.

# 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
## 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.

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

- Books which received higher number of ratings had lower average ratings than the average rating for all the books.
- When a book has a large number of ratings, it is likely to have some of the lower ratings, which drags down the average rating.

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

# Feature Engineering and Text Preprocessing

- Calculated the average review score and the total number of ratings for each book and columns added to the Books data.
- Converted the publishedDate column in the books data to date/time in the format of 4-digit year.
- Combined the categories and description columns and created a description_categories column. This column was used for NLP processes to produce the book recommendations.
- Removed special characters and numbers from the description_categories and converted text to lower.
- Tokenized the text in the description_categories column to separate each word in the text using word_tokenize from nltk library.
- Lemmatized each word using WordNetLemmatizer from the nltk library to reduce variant forms of words to one word.
- Removed stop words (the, and, are, is …) using the nltk.corpus to generate stop words list.
- I joined the words back together to form one long string for the categories_description value of each book.
- Finally , I took a subset of the data (books with >10 ratings, about 30,000 books) for the modeling.

# Making the Recommendations

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

# Model 1 - CountVectorizer and Cosine Similarity

Steps taken:

1. Used CountVectorizer from the Scikit Learn library to create vectors for the description_categories text.
2. Created the Cosine Similarity matrix for the whole data using Scikit Learn's Cosine Similarity function.
3. 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 with this model:

1. Vectorized the description_categories column using nlp.
2. Computed the similarity score of the chosen book with all of the books, including itself using Spacy.
3. Sorted the similarity scores in descending order to find the 5 most similar books to the chosen book.
4. The system computes similarity scores of a chosen title when needed instead of calculating similarity scores of the whole data in advance. With this model, this step runs fast and doesn't negatively affect the speed of the model.

# Model 3 - with SBERT Sentence Transformer and Cosine Similarity

Steps taken:

1. Created sentence embeddings for the description_categories column using SBERT SentenceTransformer.
2. Created the cosine similarity matrix 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 sorted that row and take the top 5 books with the highest similarity scores with the chosen book.
4. Creating the sentence embeddings and similarity matrix takes some time to run. However, once the matrix is created, making recommendations is very fast.

# How About Accuracy Matrix?

- Used Discounted Cumulative Gains (DCG) to measure and compare performances of the three models. Accuracy comparison table in the next slide.
- Manually compared description of chosen titles and recommended titles to determine relevancy score.
- DCG = Relevancy Score/(LOG(Recommendation Rank+1))
- Formula Source: Towards Data Science, An Exhaustive List of Methods to Evaluate Recommender Systems

# Accuracy Measure: Discounted Cumulative Gains (DCG)

## Scoring Key:

Most relevant => 2

Somewhat relevant => 1

Least relevant => 0

| | | Count Vectorizer | Count Vectorizer | Gensim / Spacy | Gensim / 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** |

# Choosing Model 1, Model 2 or Model 3?

Chosen model - SBERT Sentence Transformer pretrained model (Model 3)

1. The Spacy model (Model 2) - simpler model, runs fast, but seems to produce lower quality recommendations.
2. Count Vectorizer and Cosine Similarity model (Model 1). The model takes a while to run, but produces decent recommendations.
3. SBERT Sentence Transformer model (Model 3) The model doesn't take too long to run and seems to produce the best quality recommendations.
4. SBERT Model takes into consideration the semantic and contextual meanings of words and sentences.
5. 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

- With the reviews data containing more than 2 million rows, it requires extremely large amount of memory to process.
- Therefore, I decided to base my recommendations on just the categories and description columns of the books data at this time.

# 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 which I haven't used much. 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 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

# 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 and made them available online. 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.