WQD7001 PRINCIPLES OF DATA SCIENCE 1/2022/2023

BOOK RECOMMENDATION SYSTEM USING CONTENT-BASED FILTERING ALGORITHM

GROUP ASSIGNMENT 2

PDS G3(RL) – Group 1		
No.	**MATRIC ID**	**NAME**
1	22056764	Devayani A/P Balkrishnan
2	S2195613	Li Xin Qi
3	S2155659	Lim Yu Xuan
4	S2192763	Navaneeta A/P P Shanmugam

TABLE OF CONTENTS

1 Project Background

Systems that make recommendations to users based on a variety of parameters are known as recommender systems. These systems forecast the product that customers will be most interested in and likely to buy. This project focuses on building a recommender system for books and is named “What’s Next, Shakespeare?”. The need for this product comes from the current buying trend that has evolved during the modern era where consumers would rather buy items online than going out due to its convenience. This buying pattern has further been pushed into becoming the new norm in the market environment by the COVID-19 pandemic. In addition, books were the chosen media because unlike books, mediums such as music and movies have established household names when it comes to recommenders (i.e., Spotify and Netflix).

There are multiple approaches that can be used to make recommender systems, depending on the scope of the project and the data that is accessible. Collaborative filtering uses similarities between users and items simultaneously to provide recommendations. This allows for serendipitous recommendations; that is, collaborative filtering models can recommend an item to user A based on the interests of a similar user B. Association rule mining finds interesting associations and relationships among large sets of data items. This rule shows how frequently an itemset occurs in a transaction. On the other hand, content-based filtering uses item features to recommend other items like what the user likes, based on their previous actions or explicit feedback. This model is advantageous as it doesn't need any data about other users, since the recommendations are specific to this user. This makes it easier to scale to many users. The model can capture the specific interests of a user, and can recommend niche items that very few other users are interested in. This project aims to build a book recommendation system through content-based filtering algorithm. This system serves the purpose of narrowing down the options available to users by predicting and making suggestions on books that they may be interested in. Our project also employs natural language processing (NLP), which is a computer program's capacity to comprehend natural language, or human language as it is spoken and written.

The data used to model the book recommender is obtained by conducting web scraping on Google Books. Google Books API was employed to perform full-text searches and retrieve information such as publisher, author, viewability, eBook availability, etc. The primary resources used for the data pre-processing, EDA, and modelling are Jupyter Notebook (used to deploy Python) and GitHub (used for code sharing). As for the reproducibility of the research, a recommender system's transparency may be hampered by several design and implementation decisions, making it nearly impossible to reproduce the stated settings and insights. These

decisions span from the use of various model parameters to the way particular system components are implement (Bellogín & Said, 2021).

A significant problem of information overload that prevents timely access to things of interest on the Internet has been caused by the exponential development in the volume of digital information available and the number of Internet users. This issue has been largely resolved by information retrieval systems like Google, DevilFinder, and Altavista, but prioritization and personalization (where a system matches accessible content to a user's interests and preferences) of information were lacking (Isinkaye et al., 2015). As a result, recommender systems are more in demand than ever. By selecting important information fragments from a large volume of dynamically created material based on the user's choices, interests, or observed behavior about the item, recommender systems are information filtering systems that address the issue of information overload. Based on the user's profile and a set of criteria, recommender systems can determine whether a specific user will favor an item or not.

Users can find books, news, movies, online courses, and research articles with the use of a personalized recommendation system. One of the issues with the current recommendation system is the inaccuracy in determining user's book preference due to vulnerability of data quality. An accurate recommendation system relies heavily on sufficient data to be able to churn out high quality data for analysis and prediction. Besides that, with the exponentially increase volume of books found online, overflowing information has pushed towards users and it becomes extremely challenging for them in finding relevant books. This has also led to problems with overwhelming options, causing trouble for users in making purchasing decisions. Moreover, some of the current recommendation systems are not built according to user preference but focused on the sellers' interests, which is to increase their book sales, resulting in redundancy and irrelevant recommendation to users. By building a book recommendation system with high quality data, it can help users by recommending the right choice according to their preference. Therefore, building a book recommendation system with high quality data benefits the user.

In actuality, both service providers and customers can benefit from recommender systems. They lower the transaction costs associated with locating and choosing products in an online buying setting. It has been shown that recommendation systems enhance the effectiveness and process of decision-making. Because they are efficient ways to sell more things, recommender systems increase revenues in an e-commerce environment. Recommender systems in libraries assist users by enabling them to go beyond catalogue searches. Therefore, it cannot be

overstated how important it is to deploy effective and accurate recommendation processes within a system that will offer users reliable and relevant recommendations.

In this project, we have identified the problems of online users having difficulty in identifying relevant books with the vast e-books available online. Therefore, our goal of this proposed study is to build a recommendation system through content-based filtering algorithm in order to ease online users by predicting and suggesting the books that they may be interested in. From the bookseller's perspective, our system would be able to recommend their books to the right users, resulting in profit-gaining. We have chosen Google Books, one of the e-book platforms that is available for web scrapping. For this project, over 112,000 rows of data were obtained via web scrapping using API. However, multiple complications were encountered due to the sheer amount of data, thus only a subset of the data was employed. Out of the original datasets, only the books from the Top 10 genres were used, leading to about 36,000 rows of data. The scope of this study has limited English language books as default language.

2 Research Questions

1. Which media does not have commonly used recommendation systems?
2. Which factors can be deemed relevant to narrow down recommendations to users?
3. Which method of obtaining data will give the widest variety of books to improve data quality?
4. Which approach will be best suited based on the type of data collected?
5. How can a balance be struck between sellers' and consumers' interest by the platform?

3 Problem Objectives

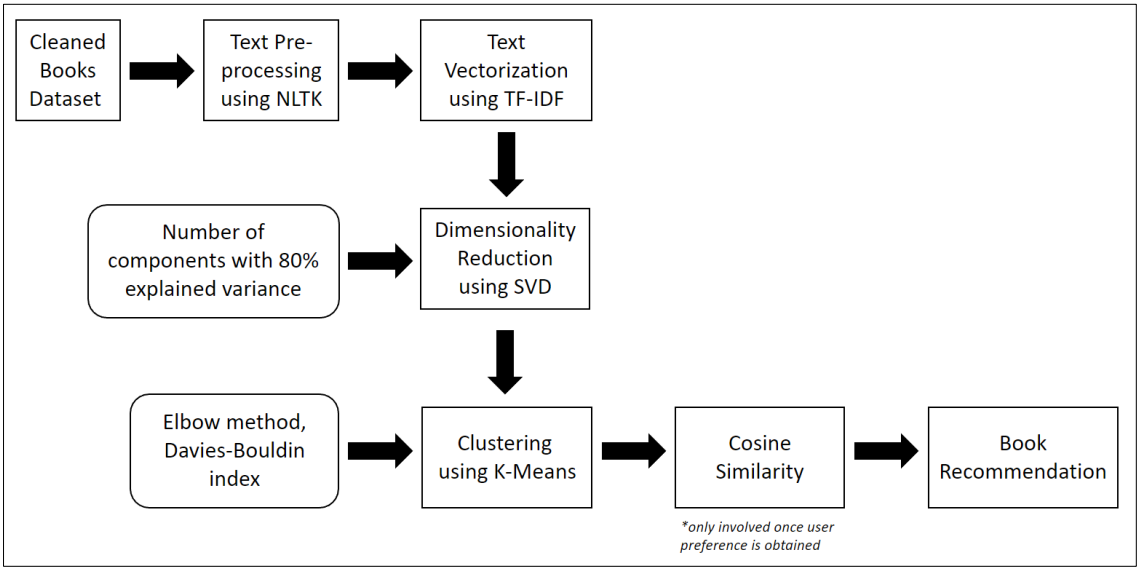
In this project, our aims are to:

a) Explore the best avenue of data collection which will result in a wide variety of books
b) Determine the key features of relevancy that will yield personalized recommendations to users
c) Evaluate different algorithms and techniques to build the content-based filtering book recommendation system
d) Deploy the content-based filtering book recommendation system in Streamlit App

4 Data Modelling

Model is the fourth stage in the OSEMN framework, where data model is built after exploring and analyzing the data. A content-based book recommendation system based on a natural language processing (NLP) approach has been used in this project. No user data is involved using this approach, instead features such as title, description and categories are used. The diagram below outlined the detailed flow in the model building stage and each modelling steps are discussed thoroughly below.

Figure 1 Proposed Modelling Framework



The cleaned book dataset is used as the data source for this project, and due to computational limits, only books from the top 10 genres are obtained and applied in the recommender system. There are around 36k of the books that belong to the top 10 categories, which make up around half of the original dataset. The texts that describe the books dataset are pre-processed using the NLTK library in Python, then converted to vector representation using a text vectorization method named TF-IDF. Dimensionality reduction is required to reduce the TF-IDF matrix before proceeding to model building, and the number of components is identified using the elbow method. The books are clustered into different segments using K-Means, and several methods like elbow method and Davies-Bouldin index are applied to find the optimal k value for the K-Means algorithm. The cosine similarity of the vectors in each cluster is calculated, and books within the same clusters and with the highest cosine similarity will be recommended.

4.1 Text Pre-processing

Prior to model building, book features such as the title, description and categories / genre are combined in a new column while the rest of the features are removed as it is not required to build an NLP based book recommender system. Any missing values in book title, description and categories are replaced with empty spaces, then concatenated into a new column named “text”.

The text pre-processing is necessary in NLP as the raw text data often contains unwanted, unimportant or missing information, which further leads to inaccurate and not reliable results. Therefore, to overcome such issues, the texts in the "text" column are preprocessed following the steps below. There are several widely used libraries and tools in Python to perform text pre-processing, like Natural Language Tool Kit (NLTK), Gensim and Sci-Kit Learn. NLTK is applied in text pre-processing task in this project.

1. Punctuations removal

   All the punctuation in the texts is removed by filtering out those characters that are in the pre-defined punctuation list in the string library in python.

2. Lower casing

   All the letters are then converted into lower case letters in order to maintain the consistency flow.

3. Tokenization

   Tokenization is used to break the context down into a smaller unit, in this case the texts are tokenized into words using NLTK Word Tokenize, and each of the words are treated as discrete elements.

4. Stop words removal

   The English stop words which carry less, or no meanings have been removed from the texts by removing the stop words listed in the NLTK library.

5. Stemming

   Stemming is a text standardization step where the suffixes or prefixes of the word are removed, and the word is stemmed into its root form. It normalizes text and reduces redundancy as most variations of word brings similar meanings. The Porter Stemmer in NLTK is used as the stemming algorithm in this project.

6. Lemmatization

   In contrast to stemming, lemmatization applies morphological analysis to words and removes only the inflectional ending in the words, which stems the word into its root form without losing its original meaning. WordNet Lemmatizer in NLTK is applied in lemmatization task in this project, which links words into its semantic relations.

4.2 Text Vectorization

There are numerous approaches to perform text vectorization, which is to convert text to numerical vectors for modelling purpose, such as One-Hot Encoding (OHE), Count Vectorizer, Bag-of-Words (BOW), N-grams and Term Frequency-Inverse Document Frequency (TF-IDF). TF-IDF is a better text vectorization approach in this task, as it provides a more compact and dense text representation as compared to OHE, which tends to create an extremely high dimensional matrix. Apart from that, the TF-IDF also weights the word according to the rarity and commonness of the word in the document, and this is not considered in other text vectorization approaches like OHE, BOW and Count Vectorizer.

Therefore, in this project, TF-IDF is selected as the text vectorization scoring schemes, as it mathematically eliminates naturally existing words in the English language and instead chooses words that are more descriptive of your content by giving a term a value based on its importance in a document scaled by its importance across all documents in your corpus. It measures how important a term is within a document relative to a corpus, while in this case, it shows the calculation of how relevant a word is within a book relative to a collection of books. It assigns weights to each word in a book based on the TF and IDF score, where higher score indicates that the word is more significant in a book and collection of books. As the name suggests, TF-IDF score is computed by multiplying the TF and IDF score using the TfidfVectorizer in sklearn module in Python.

Term Frequency (TF)

The term frequency (TF) refers to the frequency of a particular term in a book compared to the total number of words in a book. The higher the TF score of a word, the more important the term in a book.

$$Term\ Frequency\ = \frac{frequency\ of\ the\ term\ in\ the\ book}{total\ number\ of\ terms\ in\ the\ book}$$

Document Frequency (DF)

The document frequency (DF) indicates the number of books in the collection of books that contain the specific term. High DF shows that the term appears many times in many books, thus terms with high DF are not important and contribute limited meanings and values to the text analysis.

$$Document\ Frequency\ =\ number\ of\ books\ that\ contains\ the\ term$$

Inverse Document Frequency (IDF)

The relevance of words in all documents is determined using the inverse document frequency (IDF), which is the inverse of the DF. IDF reflects the proportion of books in the books selection that contains the specific term, therefore high IDF value indicates uncommon terms in the book collections and highly relevant to the book.

$$Inverse\ Document\ Frequency\ =\ \log\left(\frac{total\ number\ of\ books\ in\ the\ books\ collection}{number\ of\ books\ that\ contains\ the\ term}\right)$$

The TF-IDF matrix is computed using the TfidfVectorizer in the sklearn library in Python, and the max_df, min_df and the max_features are set to reduce the dimension of the matrix, thus prevent memory limit error in model building. However, after limiting the number of features, the matrix is still too huge to fit inside the K-Means model, therefore the dimensionality reduction is introduced before building the model.

4.3 Dimensionality Reduction

Due to the complexity of the TF-IDF matrix, the dimensionality reduction which converts the high dimensional data into a lower-dimensional subspace while preserving its essential attributes is employed. The truncated singular value decomposition (SVD) of the TF-IDF matrix is employed in this project. The truncated singular value decomposition (SVD) was chosen as the dimensionality reduction technique since its major purpose is to approximate the original data by lower-dimensional subspace, while preserving as much of the original structure as possible. The formula of the truncated SVD is shown below

$$H_{approx}\ =\ U_k S_k V_k^T$$

where U_k is a matrix with the first k columns of the matrix U , S_k is a diagonal matrix with the k largest singular values, and V_k^T is a matrix with the first k rows of the matrix V^T.

4.4 Clustering

Unsupervised learning algorithm, K-Means Clustering divides the unlabeled dataset into various clusters. A cluster is a group of data items that have been combined due to their shared characteristics. K in K-means refers to the number of centroids where a centroid is the imaginary or the real location which represents the center position of the cluster. The K-means algorithm finds k centroids, keeps the centroids as small as feasible, and then assigns each data point to the closest cluster. The means in K-means refers to the averaging of the data which is eventually finding the centroid. Data mining's K-means technique uses an initial set of randomly chosen centroids as the starting points for each cluster to process the learning data, and iterative (repetitive) calculations are then used to optimize the positions of the centroids. When the centroids have stabilized or the predetermined number of iterations has been reached, it stops building and optimizing clusters.

For this project, the K-means centroid was selected using the KElbowVisualizer class from the YellowBrick library. The elbow point on the plot represents the number of clusters where the rate of decrease in within-cluster distance slows down. From the plot, the elbow is at k = 18. Therefore, there are 18 clusters in this model.

4.5 Cosine Similarity

The cosine similarity index calculates how similar two vectors in an inner product space are to one another. It establishes whether two vectors are roughly pointing in the same direction by calculating the cosine of the angle between them. In text analysis, it is frequently used to gauge document similarity. The equation of the cosine similarity is

$$similarity = \cos(\theta) = \frac{A.B}{|A||B|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$

Regardless of the size of the documents, cosine similarity is a metric used in natural language processing (NLP) to gauge how similar they are. The cosine of the angle formed by two vectors projected in a multi-dimensional space is calculated by this cosine similarity. Because of their size, the two comparable documents may still be separated by a smaller angle even though they are far apart by the Euclidean distance. This is why the cosine similarity is helpful. The greater the similarity, the smaller the angle.

5 Data Interpretation

The last step of the OSEMN framework is the Interpret stage, where the result will be evaluated and interpreted, then put into good use. In this project, the final data product is the book recommendation system, where user can interact using the web application by simply selecting some book genres and authors that they preferred, and the top 10 recommended books will appear in the web application for user reference. The method applied in the model evaluation and the overall workflow of the book recommendation system will be discussed in detailed below.

5.1 Evaluation

There are several metrices involved in evaluating the unsupervised clustering model deployed in the modelling stage, such as the elbow method, Silhouette Score, Calinski-Harabasz index and Davies-Bouldin index. For our project, we have chosen the elbow method and Davies-Bouldin index.

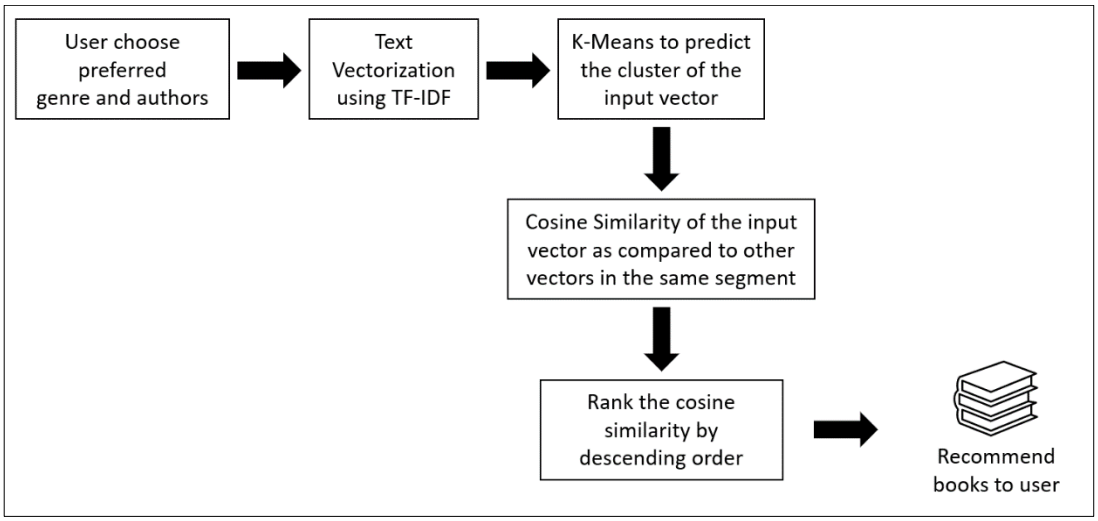
The Davies-Bouldin index is one of the metrics used to evaluate clustering algorithms. It is most frequently used to gauge how well a K-Means clustering algorithm splits data for a specific number of clusters. It is defined as

$$DB = \frac{1}{n}\sum_{i=1}^{n} \max_{j \neq i} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right)$$

where *n* is the count of clusters and *σi* is the average distance of all points in cluster *i* from the cluster center *ci*. The average similarity of each cluster to the cluster that is most like it is used to construct the Davies-Bouldin index. The clusters' separation and compactness are both captured by the DB index. This is because the measure's 'max' statement selects values repeatedly where the average point is farthest from its center and where the centers are closest to one another. The more distinct the clusters are from one another, and the better the clustering's outcome, the lower the Davies-Bouldin index (measure of average similarity) is. The Davies-Bouldin index obtained for this model is 4.7713, which is low as compared to other k values, thus it is proved that choosing 18 clusters is the best for this model.

5.2 Book Recommendation

Figure 2 The Book Recommendation System



The following discussed the background process of the book recommendation system proposed in this project.

1. Users choose their top 5 preferred genre and authors at the Book Recommendation System page on the Streamlit web application.
2. The selected genre and authors will group together, then vectorize into matrix using the same TF-IDF Vectorizer used while deploying the model.
3. The TF-IDF matrix will then input into the pre-trained SVD and K-Means model, then the model will predict the cluster based on the input.
4. The books that are tagged as the same cluster with the user input will be filtered, and the cosine similarity of the input and the books within the same cluster will be calculated.
5. The cosine similarities are ranked in descending order, then the top 10 will output to the user in the Streamlit application as a dataframe.

6 Deployment of Data Product

Lastly, the model built was deployed in a production system to ensure sustainability and create value over time. In this project, the deployment of our book recommendation system is built in Streamlit application. The following details show the steps on deploying our data product:

1. **Development Environment**

   Prior to that, a development environment is set up. This includes:

   a. installing conda
   b. creating new conda environment
   c. activating conda environment

d. install Streamlit package

2. **Integrated Development Environment**

   A Python file is created in Integrated Development Environment. For this project, Visual Studio Code is used.

3. **Import Library**

   Necessary libraries are imported, such as streamlit, pandas, numpy, matplotlib etc.

4. **Load data**

   Dataset that are used in our project are loaded to be readable in the python file.

5. **Build Dashboard**

   Our Streamlit app is split into three sections via the Navigation bar, which consists of:

   a. **Project Information**

      Project Information is shown as the homepage. A brief explanation on our project background and objectives on building this book recommendation system are stated as an introduction to audience.

   b. **Book Data Facts**

      Under this section, descriptive statistics on the dataset available are demonstrated in graphs and charts form. It describes the book attributes such as genres, authors, price, availability mode, ratings etc. Top popularity books and authors are illustrated as well.

   c. **Book Recommendation Engine**

      There will be 2 questions to ask users on their top 5 preferred genres and authors. Various Streamlit widgets such as dropdown, multiselect etc. are used to gather input from the user. The user is first asked to select their top 5 preferred genres. If condition function is used to ensure that the second question, i.e. the preferred authors available for selection are based on the preferred genre given.

      The given inputs are used to query the model built and the recommendation system will generate personalized outcome.

6. **'streamlit run' Command**

   Save the latest change made in the Python file and run the above command on local machine. The data product is now available on Streamlit App for deployment.

Images of the deployed streamlit app can be viewed in the appendix segment. All the code and the app can be found in our GitHub repository at https://github.com/xinqili98/wqd7001grp1.git.

7 Insights and Conclusion

There are several challenges and limitation faced in this project. In the Obtain stage in the OSEMN model, the dataset scrapped on Google Books has a limitation on data privacy protection and thus lack of user dataset and ratings. Moving towards the modelling and deployment phase, huge dataset is the main challenge encountered. The problem has causes long processing time for modelling phase due to the matrix size. This has also led to failure in uploading to GitHub and reading dataset in Streamlit during the deployment phase.

As for our insights, from a business standpoint, the webapp can be sold to retailers as tools to help their customers and give them an upper hand compared to other book sellers. Their inventory of books can also be pushed into the model to further tailor/personalize the product to each retailer. Furthermore, if we were to change the data pushed into the model from books to scientific literature/journals, this could become a very useful tool for researchers working on their projects/thesis. Literature review takes up a large portion of a researcher's time and energy. This tool could cut down on their time spent looking for works that are similar to their topic of interest.

On the subject of reproducibility, our recommendation algorithm is reproducible by using the same data and methods. The dataset scrapped from Google Books for this project can be shared to the public to replicate the results using the same data. The only downside is that if the data was newly scrapped from Google Books, the project will be replicable and not reproducible as the scrapped book data will change on the basis of time.

To conclude, all facets of the OSEMN framework were observed during this project. A content-based book recommendation system was successfully built and deployed, and we hope it will be beneficial to both service providers and consumers.

8 References

Bellogín, A., & Said, A. (2021). Improving accountability in recommender systems research through reproducibility. User Modeling and User-Adapted Interaction, 31(5), 941–977. https://doi.org/10.1007/s11257-021-09302-x

Deepanshi. (2022, December 5). Text preprocessing NLP: Text preprocessing in NLP with python codes. Analytics Vidhya. Retrieved January 26, 2023, from https://www.analyticsvidhya.com/blog/2021/06/text-preprocessing-in-nlp-with-python-codes/

Dixit, S. (2021, October 7). Content based music recommendation system. Medium. Retrieved January 27, 2023, from https://medium.com/@shashwatdixit6311/content-based-music-recommendation-system-9db1245a04ea

Fatih Karabiber Ph.D. in Computer Engineering, Fatih Karabiber Ph.D. in Computer Engineering, Psychometrician, E. R. L., & Editor: Brendan Martin Founder of LearnDataSci. (n.d.). TF-idf - term frequency-inverse document frequency. Learn Data Science - Tutorials, Books, Courses, and More. Retrieved January 26, 2023, from https://www.learndatasci.com/glossary/tf-idf-term-frequency-inverse-document-frequency/

GeeksforGeeks. (2023, January 19). Understanding TF-IDF (term frequency-inverse document frequency). GeeksforGeeks. Retrieved January 26, 2023, from https://www.geeksforgeeks.org/understanding-tf-idf-term-frequency-inverse-document-frequency/

Google. (2016, March). Getting Started | Google Books APIs |Google Developers. Google Developers. https://developers.google.com/books/docs/v1/getting_started

Google. (n.d.-a). About Google Books – Google Books. Www.google.com. https://www.google.com/intl/en/googlebooks/about/index.html

Google. (n.d.-b). Class: Google::Apis::BooksV1::Volume. Retrieved December 8, 2022, from https://googleapis.dev/ruby/google-api-client/v0.40.0/Google/Apis/BooksV1/Volume

Google. (n.d.-c). Collaborative Filtering | Recommendation Systems. Google Developers. https://developers.google.com/machine-learning/recommendation/collaborative/basics

Google. (n.d.-d). Content-based Filtering | Recommendation Systems. Google Developers. https://developers.google.com/machine-learning/recommendation/content-based/basics

Google. (n.d.-e). Volume | Google Books APIs. Google Developers. https://developers.google.com/books/docs/v1/reference/volumes

Goyal, C. (2021, June 22). Text vectorization and word embedding: Guide to master NLP (part 5). Analytics Vidhya. Retrieved January 26, 2023, from https://www.analyticsvidhya.com/blog/2021/06/part-5-step-by-step-guide-to-master-nlp-text-vectorization-approaches/#h2_10

Hou, D. (2022). Personalized Book Recommendation Algorithm for University Library Based on Deep Learning Models. Journal of Sensors, 2022, 1–6. https://doi.org/10.1155/2022/3087623

Ijaz, F. (2020). Book Recommendation System using Machine learning. Easychair.org. https://easychair.org/publications/preprint/Ks4N

Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. A. (2015). Recommendation systems: Principles, methods and evaluation. Egyptian Informatics Journal, 16(3), 261–273. https://doi.org/10.1016/j.eij.2015.06.005

Khan, B. M., Mansha, A., Khan, F. H., & Bashir, S. (2017). Collaborative filtering based online recommendation systems: A survey. 2017 International Conference on Information and Communication Technologies (ICICT). https://doi.org/10.1109/icict.2017.8320176

Lau, D. C. H. (2019, January 10). 5 Steps of a Data Science Project Lifecycle. Medium. https://towardsdatascience.com/5-steps-of-a-data-science-project-lifecycle-26c50372b492

Lazy2choose. (2021, August 1). Text data pre-processing: Why must text data be pre-processed ? Analytics Vidhya. Retrieved January 26, 2023, from https://www.analyticsvidhya.com/blog/2021/08/why-must-text-data-be-pre-processed/

Mathew, P., Kuriakose, B., & Hegde, V. (2016). Book Recommendation System through content based and collaborative filtering method. 2016 International Conference on Data Mining and Advanced Computing (SAPIENCE). https://doi.org/10.1109/sapience.2016.7684166

Puritat, K., Julrode, P., Ariya, P., Sangamuang, S., & Intawong, K. (2021). Book Recommendation for Library Automation Use in School Libraries by Multi Features of Support Vector Machine. International Journal of Advanced Computer Science and Applications, 12(4). https://doi.org/10.14569/ijacsa.2021.0120426

Qodsi, M. (2020, August 3). Higher accuracy and less process time in text classification with LDA and TF-IDF. Medium. Retrieved January 27, 2023, from https://towardsdatascience.com/higher-accuracy-and-less-process-time-in-text-classification-with-lda-and-tf-idf-d2d949e344c3

Sarma, D., Mittra, T., & Shahadat, M. (2021). Personalized Book Recommendation System using Machine Learning Algorithm. International Journal of Advanced Computer Science and Applications, 12(1). https://doi.org/10.14569/ijacsa.2021.0120126

Sawant, M. (2019, July 9). Truncated singular value decomposition (SVD) using Amazon Food Reviews. Medium. Retrieved January 27, 2023, from https://medium.com/swlh/truncated-singular-value-decomposition-svd-using-amazon-food-reviews-891d97af5d8d

Sohail, S. S., Siddiqui, J., & Ali, R. (2013, August 1). Book recommendation system using opinion mining technique. IEEE Xplore. https://doi.org/10.1109/ICACCI.2013.6637421

Tewari, A. S., & Priyanka, K. (2014). Book recommendation system based on collaborative filtering and association rule mining for college students. 2014 International Conference on Contemporary Computing and Informatics (IC3I). https://doi.org/10.1109/ic3i.2014.7019651

Tewari, A. S., Kumar, A., & Barman, A. G. (2014, February 1). Book recommendation system based on combine features of content based filtering, collaborative filtering and association rule mining. IEEE Xplore. https://doi.org/10.1109/IAdCC.2014.6779375

Tian, Y., Zheng, B., Wang, Y., Zhang, Y., & Wu, Q. (2019). College Library Personalized Recommendation System Based on Hybrid Recommendation Algorithm. Procedia CIRP, 83, 490–494. https://doi.org/10.1016/j.procir.2019.04.126

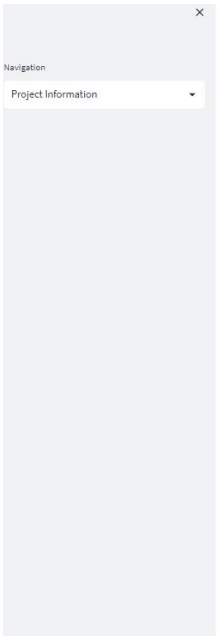
Toudeshki, F. G. (2020, April 5). Dimensionality Reduction Methods. Medium. Retrieved January 27, 2023, from https://medium.com/@farnazgh73/dimensionality-reduction-methods-d4d44f7db6b8

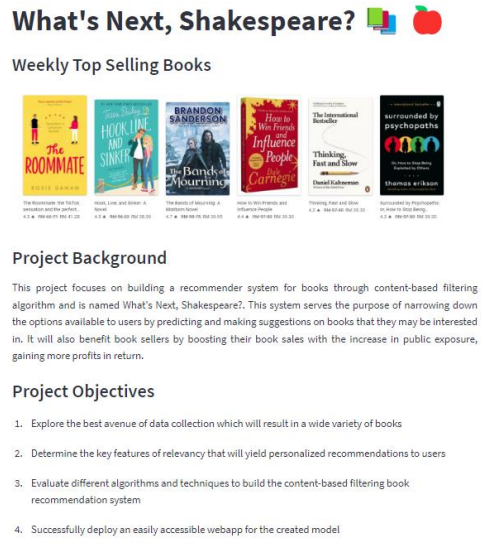
Wadikar, D., Kumari, N., Bhat, R., & Shirodkar, V. (2020). Book recommendation platform using deep learning. International Journal of Engineering and Technology, 07, 6764–6770.

9 Appendix

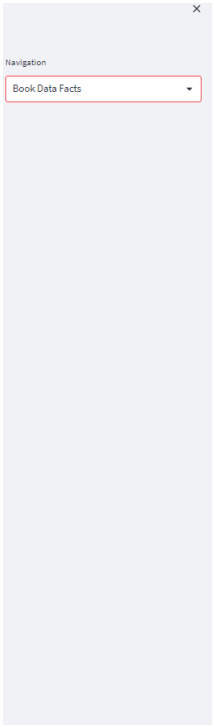
Streamlit App

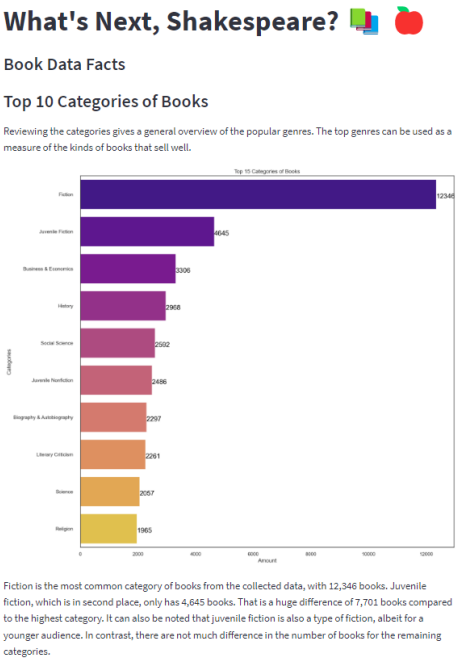
1. Project Information





2. Book Data Facts





3. Book Recommendation Engine