ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

BUSINESS ANALYTICS Master of Science

MLCA Project

TOPIC MODELLING USING NMF AND BOOKS RECOMMENDATION SYSTEM

PROJECT TEAM NAME: HOTEL

ANASTASIOS PARLANIS (Data Extraction and Web Scrapping)

ANTONIOS DIMOGLOU (AI Agent – Flask Project)

SOTIRIOS MARGONIS (NMF application – Data processing)

DIMITRIOS PALOUKTSIS (NMF application – Data processing)

Table of Contents

1 Introduction

Nowadays, book readers are facing lots of challenges in discovering books that are aligned with their interests due to the vast amount of literature that is spared online. This assignment addressed these challenges by developing a personalized book recommendation system using machine learning and natural language processing (NLP) techniques. Using the Project Gutenberg online book repository, lots of free to public books were scrapped and analyzed using the Non-Negative Matrix method (NMF), which is a common practice in topic modelling, and especially in text analysis.

As far as businesses are concerned, this recommendation system could provide a lot of value. Users might get more engaged by using this interactive and innovative recommendation system. For instance, the system could be installed in bookstores or libraries while being connected to the business' database of book availability. Then, the customers would be able to ask for a book recommendation, diminishing their time of discovering and searching through all the shelves. Hence the overall user experience and user loyalty would be improved, leading to the increase of revenue of the business. The following report entails all the steps of the development of the recommendation system, from gathering and preprocessing data to building the model and creating a web application using Flask. Its goal is to highlight the potential of machine learning in book discovery and to underline how business could benefit from it.

The structure of this report is the following: Section 2 analyzes the data that was used in this assignment and how they were gathered. Section 3 analyzes the text pre-processing that was done in order the data to be ready to be used in the model. Section 4 presents the theory of NMF, the implementation using the data that was gathered and analyzed, and it also provides the results after the implementation of the model. Section 5 presents the development process of the Flask-based project which was designed to recommend books based on user input and as well as its corresponding results. Finally, the report finishes with sections 6 and 7 in which the key challenges and the conclusion of the assignment are presented respectively.

2 Data

2.1 Data web scrapping

The books data was gathered from the Project Gutenberg using the following link https://www.gutenberg.org/ebooks/search/?sort_order=downloads. It contains a very large repository of free books in several formats. The format that was needed for this assignment was the plain text (txt.uft-8). A python code was developed for scrapping the Project Gutenberg and focusing on the most downloaded books. Downloading such a large volume of public domain books is challenging, and for this reason the Python script uses an automated approach in order to be able to collect the data efficiently. It uses the *requests* library to send HTTP requests and download HTML content from the site, while *BeautifulSoup* parses the HTML to extract book URLs. Once the URLs are identified, the script retrieves the plain text version of each book and saves it locally. It handles pagination by iterating through search result pages and ensures a structured downloading process. For handling pagination and ensuring a well-structured downloading process, the script iterates through search result pages. Additionally, the scraper uses regular expressions to identify the correct text file format and avoids overwhelming the server by implementing rate limiting between requests. Finally, 930 text files were collected.

2.2 CSV creation

After downloading the books, another Python script was written to extract key metadata like title, author, language, publication year and as well as the full text of each book that is needed in the analysis. The script iterates over the text files in the specified folder, identifies the necessary metadata by looking for specific patterns (e.g., lines starting with "Title:", "Author:", and "Language:"), and captures the main body of the text between the "START" and "END" markers of the Project Gutenberg content. The extracted data is then written to a CSV file in a structured format, with each row containing the book's title, author, language, publication year, and the full text in a single row. In general, this CSV file contains all the necessary data that will be imported later and analyzed in this assignment.

2.3 Data

There are 930 rows in the CSV file, each row for a single book. The columns are the following: ‘Title’, ‘Author’, Language’, ‘Year’ and ‘Text’, as it is previously mentioned. First of all, the vast majority of the books are written in English (*See Figure 1*), and there are the books that were kept in a subset of the data that was used in the analysis. Figure 2 shows the number of books by author in the dataset. Charles Dickens, Various, William Shakespeare and Mark Twain are the most common authors in the dataset. It is also important to note that there are 46 books for which the information about the author’ name was missing. Furthermore, most of the books were published during the decades of 2000 and 2010, as Figure 3 suggests.

Figure 1

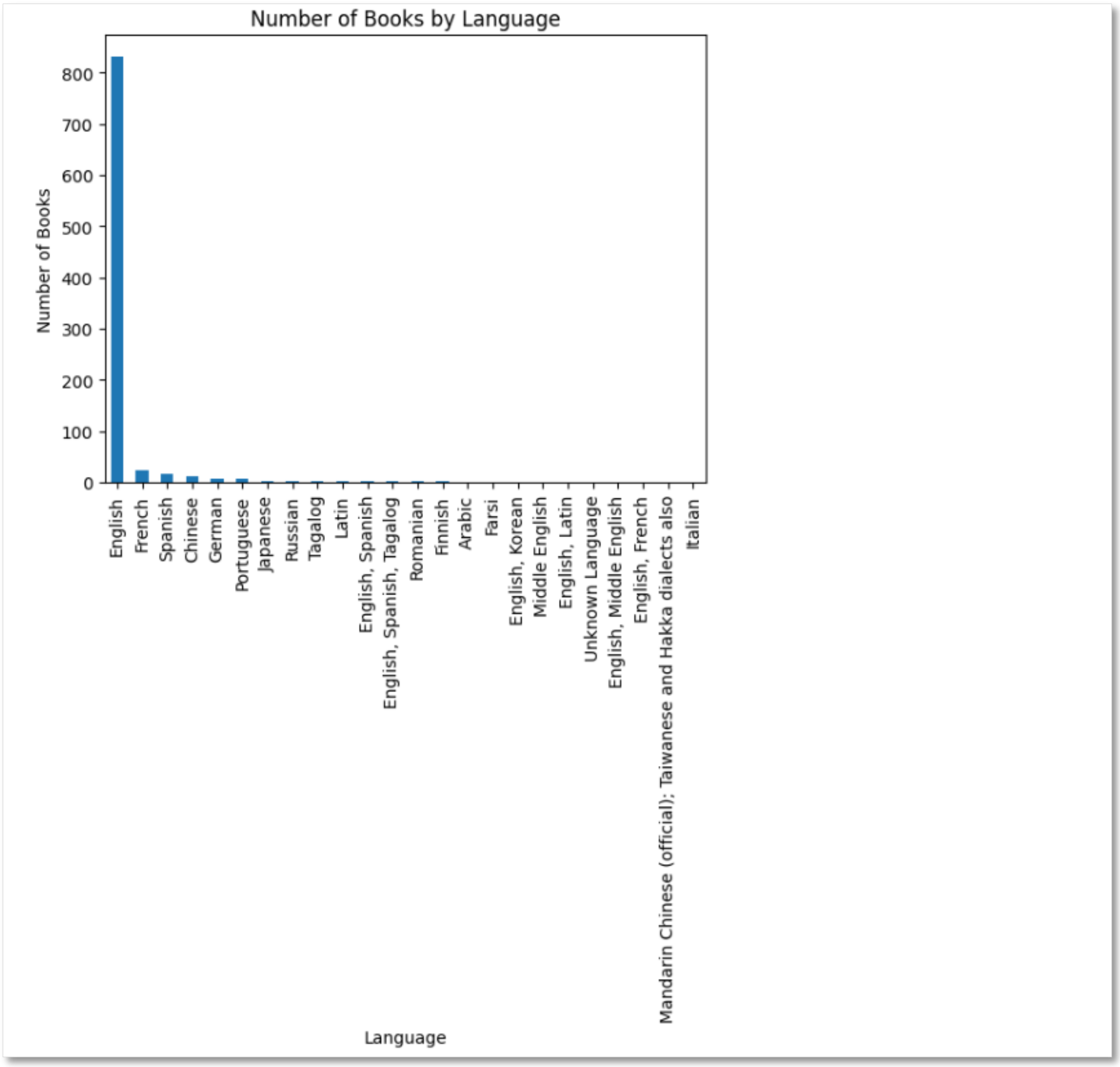

Number of Books by Language
Number of Books
800
700
600
500
400
300
200
100
0
English
French
Spanish
Chinese
German
Portuguese
Japanese
Russian
Tagalog
Latin
English, Spanish
English, Spanish, Tagalog
Romanian
Finnish
Arabic
Farsi
English, Korean
Middle English
English, Latin
Unknown Language
English, Middle English
English, French
Mandarin Chinese (official); Taiwanese and Hakka dialects also
Italian
Language

Figure 2

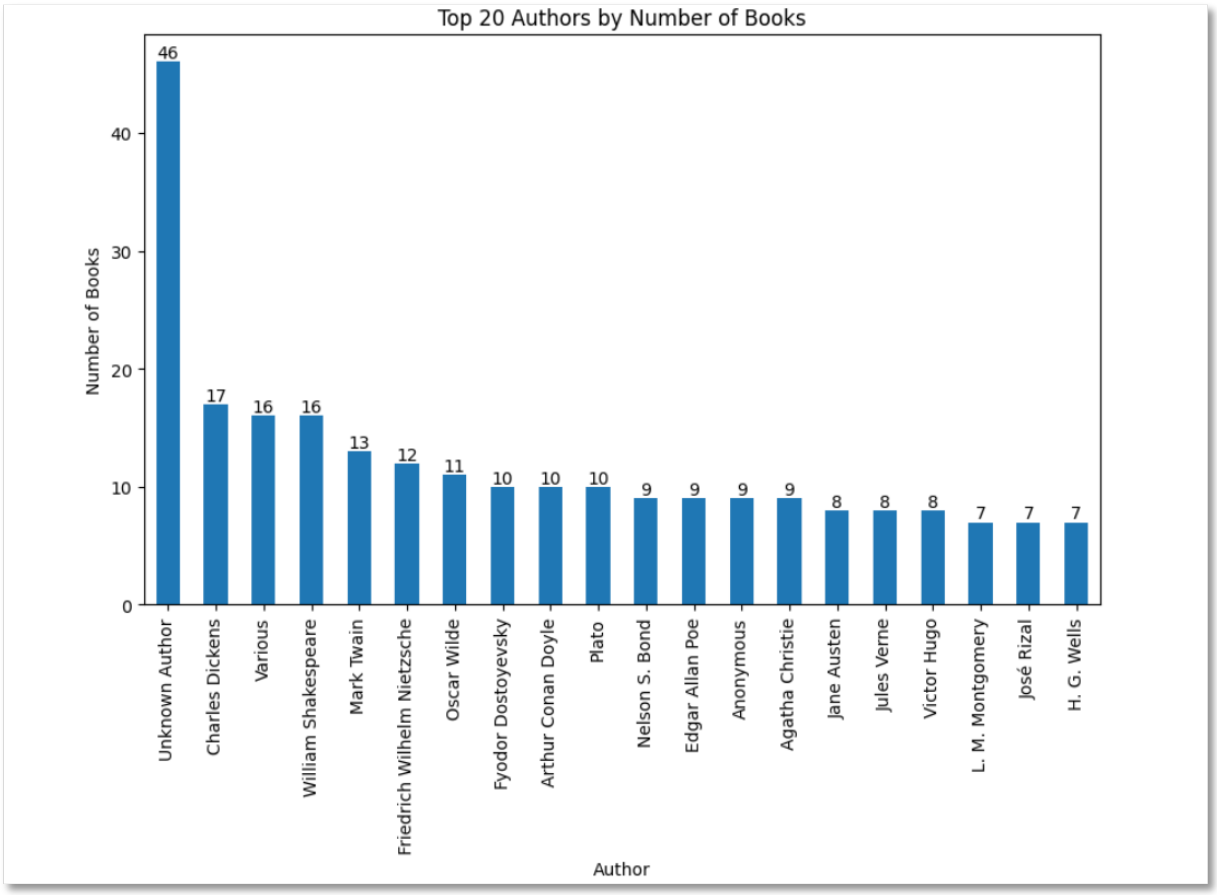
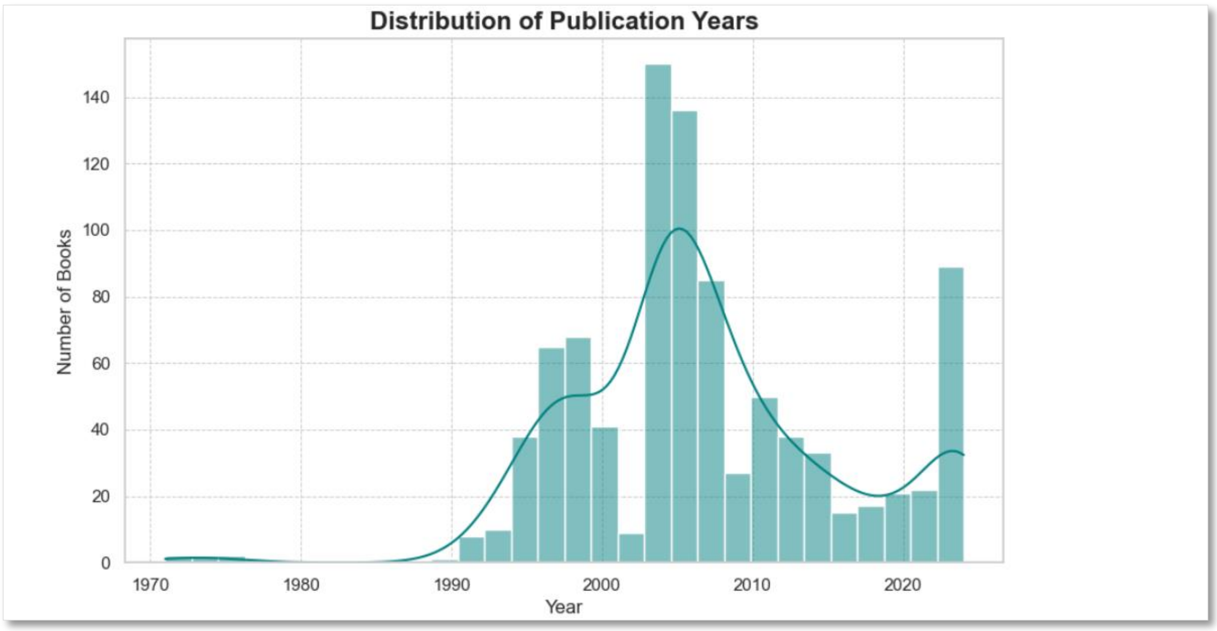


Figure 3



The length of the characters that were analyzed in this assignment can be examined in Figures 4 and 5. The histogram that is depicted in Figure 4 gives a more granular view of the distribution of text lengths. It is right skewed which means that there are lots of books of moderate length in the dataset, and very long books are much less common. It is also noted that most of the books contain around 200,000-300,000 characters, which can be interpreted as the most common length in this dataset. The concentration of books between 100,000 and 400,000 is also aligned with the pie chart in Figure 5, which also suggests that a large portion of the books falls in the range of characters between 200,000 and 800,000 (46,8%). A significant portion of the books fall in the range less than 100,000 characters (16.3%) and a very small portion contains more than 1.2 million characters (10.3%) which is also aligned with the histogram. In general, both figures state that the large share of the books fall between 200,000 and 800,000 characters, with a particular concentration in the range 200,000 to 300,000.

Figure 4

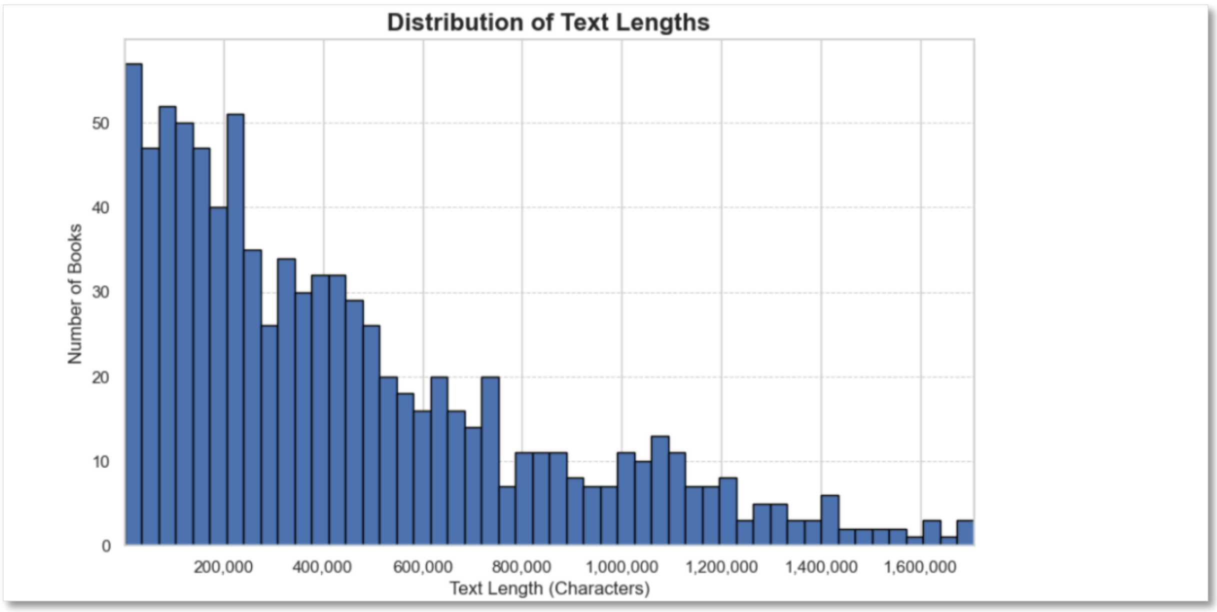
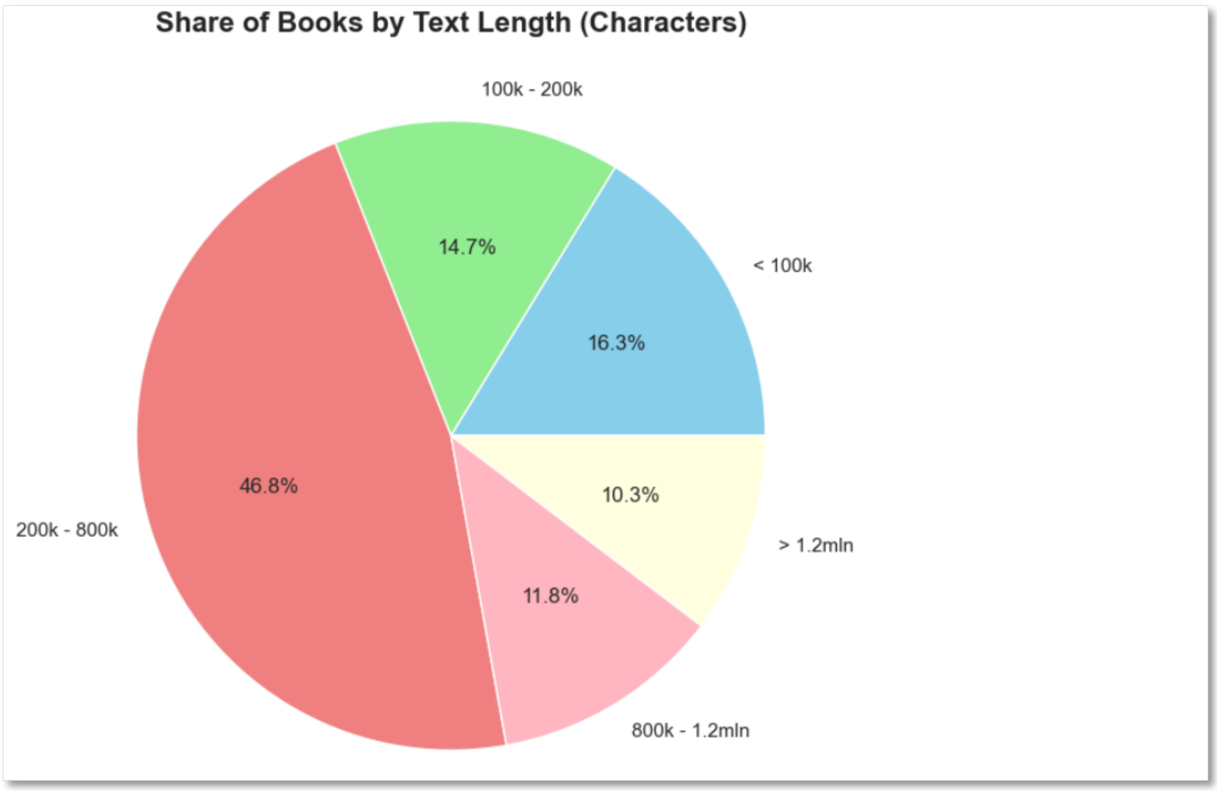
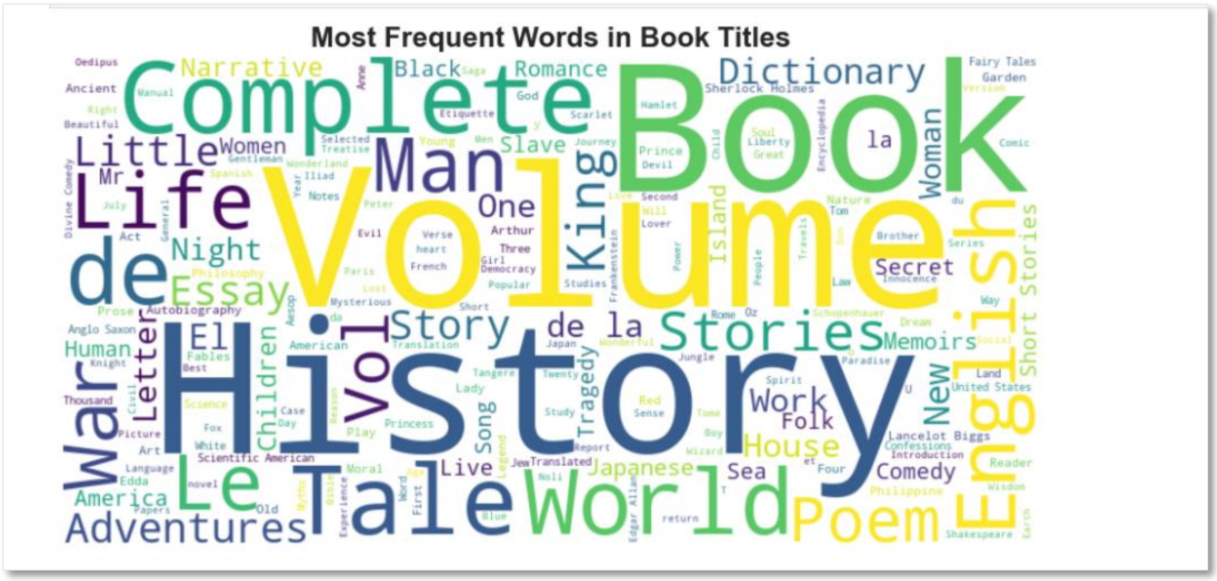


Figure 5



Finally, Figure 6 contain a word cloud. It is a visualization which shows the frequency of words. The larger the size of a word, the higher its frequency is the book titles. It could also be an initial indication of the topics that are expected to come up in the topic modelling thereafter. For instance, words like “History”, “Volume”, “Story”, “Book”, “War”, “World”, “King”, “Ancient”, “Adventure”, might suggest that the dataset contains historical books, collection of stories or adventurous ones. Words like “Poem”, and “Tale” might suggest poetry. Additionally, there are also other words like “Comedy”, “English”, “Life”, “Woman”, “Romance”, “Philosophy”, “Memoirs” which come up with a moderate size and they might also suggest that the dataset contains significant variety of topics.

Figure 6



3 Data Preprocessing

3.1 Text Pre-processing

Let's now proceed to the text preprocessing part before feeding the dataset into the model. The "tqdm" library combined with pandas just ensures that we can keep track of our process displaying a progress bar. This is useful while handling big datasets. In our case, the process takes a few minutes to run.

Next, we use a regular expression to remove punctuation from the text and remove noise from the data. Using library NLTK we load a set of stopwords in English. In general, stopwords are frequent words in a language, such as "the," "is," "in," "and" or "to." These words usually do not provide any semantic value and may create noise on the analysis, making it harder to identify key themes and specifically in our scenario, topics. By removing them, we can focus on meaningful terms and words that will contribute to our model's creation.

Making use of all the above we define a Preprocessing Function. In particular, the function executes the following steps:

- Converts the text to lowercase to follow a universal format.
- Removes punctuation using the precompiled regex.
- Tokenizes the text into individual words, limiting the number of words to 5,000.
- Filters out stopwords from the tokenized words.
- Finally, it joins the processed words back into a single string for further analysis.

Then, we run this function in our book_data DataFrame, and more specifically to the 'Text' column. A progress bar is also visible by using "progress_apply".

3.2 Theory and Application of TF-IDF (Term Frequency-Inverse Document Frequency) Vectorization

- TF (Term Frequency): counts how many times a word appears in a book.
- Inverse Document Frequency (IDF): measures how unique is a word across all books.

By using both TF and IDF we can find words that are important for a specific book but do not appear often in other books as well. For instance, consider a book having to do with biology. The word "biology" will appear many times on that book whereas few on others. However, some words that are not considered as stopwords may be present with a high frequency across many books. IDF helps us avoid giving high importance to such words. More specifically:

- max_features=5000: Limits the number of unique words to the top 5,000, focusing on the most important and relevant ones.
- stop_words='english': Removes all English stopwords.
- fit_transform: Fits the vectorizer to the data and transforms the text into a TF-IDF weighted document-term matrix.

After following the abovementioned steps, we can use the produced TF-IDF matrix as an input for the NMF model.

4 Introduction to Topic Modelling [1]

Topic modelling in simple terms is a way of identifying patterns in a large body of text. Essentially it is a method of clustering words into topics. This text analysis technique called topic modeling can be used to find hidden topics in huge document sets. By assembling words that frequently occur together into groups known as "topics," it facilitates pattern recognition. These subjects don't require any prior information or preconceived notions to highlight relationships, trends, or themes across the text. Latent Dirichlet Allocation (LDA) is a popular model for topic modeling that uses probability to assign words to themes, but this model was not used in this project. In the following units of this chapter more details will be given about the model used.

Large text collections are usually very hard to summarize and that’s where topic modelling becomes very useful in categorizing large pieces of text and extracts broader meanings as topics.

4.1 History of Topic modelling[2]

The first application of topic modelling is traced back to 1990s in disciplines such as information retrieval, statistics and machine learning. More specifically, researchers in 1990s searched for ways of organizing large

[1] Brett, M. R. (2012). *Topic modeling: A basic introduction*. Journal of Digital Humanities, 2(1). Retrieved from https://journalofdigitalhumanities.org/2-1/topic-modeling-a-basic-introduction-by-megan-r-brett/

[2] Liu L, Tang L, Dong W, Yao S, Zhou W. An overview of topic modeling and its current applications in bioinformatics. Springerplus. 2016 Sep 20;5(1):1608. doi: 10.1186/s40064-016-3252-8. PMID: 27652181; PMCID: PMC5028368.

collections of data in predefined categories. Topic modelling gained a lot of popularity with the development of Probabilistic Latent Semantic Analysis (PLSA) whose main goal was to detect relationships between words and concepts. However, a bigger success was deemed the latent Dirichlet allocation (LDA) proposed by Blei et al. (2003) is an even more complete probabilistic generative model and is the extension of PLSA. Words are assigned to themes by the probabilistic generative model LDA, which takes into account the co-occurrence of words in documents. By enabling a document to be represented as a mixture of themes, it outperformed earlier models and was more flexible and resembled the structure of natural language. LDA became the basis model for contemporary topic modeling techniques as a result of this innovation.

Besides LDA newer models have also emerged such as the Hierarchical LDA and Dynamic Topic Models (DTM), which have enhanced the methods' ability to produce topics that evolve over time or by segmenting topics into hierarchies.

4.2 Non-Negative Matrix Factorization (NFM)[3]

In this project the Non-Negative Matrix Factorization (NFM) method was employed. NFM is a statistical method that facilitates the reduction of the dimensions of the input data. Hence, makes it suitable in the context of categorizing books since they constitute inputs with very high dimensionality.

Non-Negative Matrix Factorization (NMF) is especially helpful for applications like topic modeling since it divides a non-negative matrix into two smaller non-negative matrices, which makes it excellent at extracting hidden patterns. Since NMF guarantees non-negativity, as opposed to other techniques like Latent Semantic Analysis (LSA), the factors (or components) generated are guaranteed to be easier to read because they naturally fit with real-world contexts where negative values are uncommon or illogical.

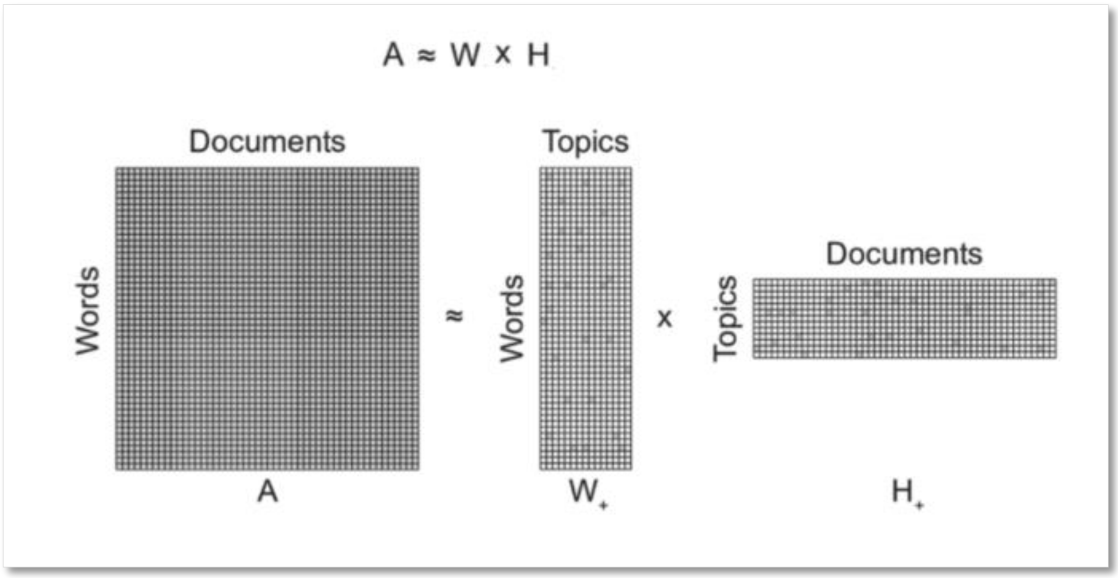
One popular application of NMF is in topic modeling, where it takes as input a term-document matrix. This matrix represents the frequency or importance of terms across multiple documents. Often, the matrix is normalized using TF-IDF (Term Frequency-Inverse Document Frequency). As depicted in Figure 7: NMF inputs and outputs the result of NMF is the decomposition of the input matrix (A) into two matrices (W and H):

- A matrix representing the relationship between terms (words) and topics (W).
- A matrix representing the relationship between topics and documents (H) .

This transformation allows us to better understand how specific words cluster around certain topics, and how these topics manifest across different documents.

[3] Chirag. (2021, June 26). *Step by Step Guide to Master NLP – Topic Modelling using NMF*. Analytics Vidhya. Retrieved from https://www.analyticsvidhya.com/blog/2021/06/part-15-step-by-step-guide-to-master-nlp-topic-modelling-using-nmf/

Figure 7: NMF inputs and outputs



The Basic Mathematical Concept:

At its core, NMF operates on the principle of matrix factorization, which is the process of decomposing a large matrix into the product of two smaller matrices. In the case of NMF:

$A \approx W \times H$

Where:

- A is the original data matrix.
- W represents the latent factors or components.
- H indicates how these factors contribute to the features.

In the context of this project:

- A (Document-word matrix): Input that contains which words appear in which documents.
- W (Basis vectors): The topics (clusters) discovered from the documents.
- H (Coefficient matrix): The membership weights for the topics in each document.

Objective Function:
The core objective of NMF is to minimize the difference between the original matrix A and the product WH. The function typically used to measure this difference is the Frobenius norm:

$\min || A - WH ||_F$

The Frobenius norm sums up the squared differences between corresponding elements of X and WH, effectively representing the error in approximation.

Optimization Methods:
There are various methods to minimize this reconstruction error:

1. Multiplicative Update Rule: A popular approach where the algorithm iteratively adjusts the values in W and H, ensuring that the matrices stay non-negative.
2. Alternating Least Squares (ALS): In this method, W and H are updated alternately by fixing one while optimizing the other.
3. Gradient Descent: This standard method adjusts both matrices step-by-step by following the gradient of the error function.

Choosing Components:
A critical part of NMF is determining the number of components r. This number defines the dimensionality of the resulting matrices. Setting r too high risks overfitting, while setting it too low could lead to a loss of essential patterns. Heuristics, cross-validation, or domain expertise are often used to make this decision.

4.3 NMF Topic Modeling implementation

With the text data vectorized, Non-Negative Matrix Factorization (NMF) was applied to uncover latent topics within the dataset. NMF is a matrix factorization technique that decomposes the TF-IDF matrix into two non-negative matrices: one representing the relationship between words and topics, and the other representing the relationship between topics and documents.

The following parameters were used for the NMF model:

- Number of Topics: The model was configured to extract 10 topics, balancing interpretability with detail in the topic representation.
- Random State: A fixed random state (random_state=42) was set to ensure reproducibility of results.
- The NMF algorithm was trained on the TF-IDF matrix, resulting in two output matrices:
  W Matrix: This matrix represents the distribution of words across topics, facilitating the identification of the most relevant words in each topic.
  H Matrix: This matrix indicates the contribution of each topic to the individual documents.

4.3.1.1 Topic Preview

To interpret the results of the NMF model, a function (get_nmf_topics) was developed to extract the top words associated with each topic. This function involves the following steps:

1. Retrieval of Feature Names:

The function retrieves the feature names (words) from the TF-IDF vectorizer, corresponding to the columns of the TF-IDF matrix.

2. Identification of Top Words:

For each topic, the function sorts the words by their importance, as determined by the values in the W matrix, and extracts the top words for each topic.

4.3.1.2 Model in practice

The next part of our project involves creating a function to search for books based on a specific word, using the NMF topic model we've already built. This function allows us to find books that have a similar topic distribution to the word we input.

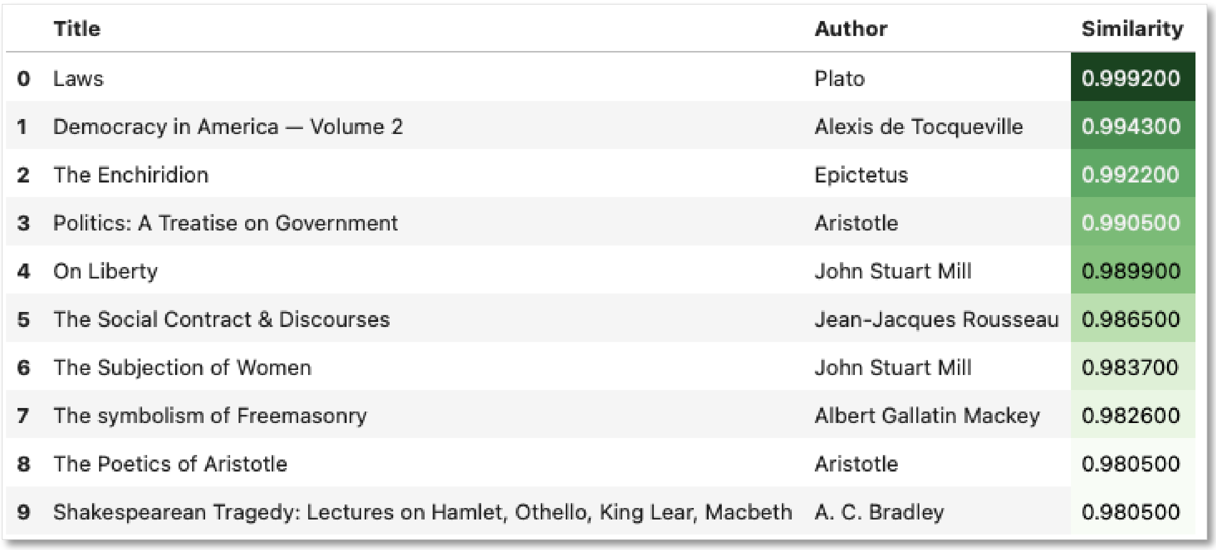
First, the word provided by the user is transformed into a format that matches the rest of the data by using the same TF-IDF vectorization process. This ensures that the word is analyzed in the same way as the text from the books, making it easier to compare them directly.

Next, the transformed word is projected into the NMF topic space, which gives us a distribution of topics for that word. This step helps us understand how the word relates to the topics we've already identified in the books. Once we have the word's topic distribution, we calculate how similar it is to the topic distributions of each book using cosine similarity. This method measures how closely the word's topics align with those in the books.

After calculating these similarity scores, they are added to the dataset, and we rank the books based on their relevance to the word. The top 10 books that are most similar are selected and displayed. To make the results easier to interpret we apply a color gradient (light green stands for lower similarity; however, it's still a recommendation, while darker green indicates a stronger recommendation / very high similarity).

In the end, the function outputs a table with the titles, authors, and similarity scores of the top 10 books that match the word entered. Figure 8 displays an example from using the word 'politics'.

Figure 8



	Title	Author	Similarity
0	Laws	Plato	0.999200
1	Democracy in America — Volume 2	Alexis de Tocqueville	0.994300
2	The Enchiridion	Epictetus	0.992200
3	Politics: A Treatise on Government	Aristotle	0.990500
4	On Liberty	John Stuart Mill	0.989900
5	The Social Contract & Discourses	Jean-Jacques Rousseau	0.986500
6	The Subjection of Women	John Stuart Mill	0.983700
7	The symbolism of Freemasonry	Albert Gallatin Mackey	0.982600
8	The Poetics of Aristotle	Aristotle	0.980500
9	Shakespearean Tragedy: Lectures on Hamlet, Othello, King Lear, Macbeth	A. C. Bradley	0.980500

5 Flask Project – Agent for Books Recommendations

This section of the report outlines the development process for a Flask-based project designed to recommend books based on user input. The project leverages natural language processing (NLP) techniques to analyze user queries and provide relevant book recommendations. It covers the step-by-step approach used to set up and run the application. The focus is on how the system processes the input, matches it with the dataset, and returns personalized book suggestions.

The Unix terminal is used to set up and run the Flask application. Specifically, virtual environment was created and activated to isolate project dependencies and avoid conflicts with other projects (python3 -m venv venv) and to install packages within it (source venv/bin/activate). Moreover, all necessary libraries and frameworks are installed using the requirements file, ensuring that the project has all the needed dependencies (pip install -r requirements.txt). Finally, the flask application is started, allowing it to handle requests and provide recommendations (python app.py). All necessary files for the development of this Flask project are included and sent separately.

5.1 Agent for Books Recommendation Analysis

The flask agent for book recommendations is the completion and final part of the project, integrating multiple components in order to create an effective and user-friendly application. The main goal is to facilitate book discovery using queries, allowing users to find relevant book titles, authors and year based on their interests. Specifically, the agent leverages a combination of natural language processing (NLP) techniques, machine learning models, and a streamlined web interface to provide users with tailored book suggestions.

5.1.1 Data Preparation

The input for the Flask Agent for Book Recommendations is based on the file 'BOOKS_DATA_updated.csv', as described in Section 2.2 of this report. This file contains essential information for each book, including the title, author, publication year, and a text description.

The Flask Agent for Book Recommendations leverages the pre-trained NMF (Non-negative Matrix Factorization) model for generating book suggestions. As outlined in Section 4.3, the NMF model was trained on the dataset's book descriptions to uncover underlying topics and classify books into thematic categories. In the Flask application, the user's query is transformed into a topic distribution that is compared against the topic distributions of the available books using cosine similarity. This method allows for highly relevant recommendations by identifying books with similar thematic structures to the user's input. The pre-trained models, including the TF-IDF vectorizer and NMF model, are loaded from serialized files (tfidf_vectorizer.pkl, nmf_model.pkl, and nmf_topics.pkl) for efficient query processing within the web interface

5.1.2 User Interaction via Flask

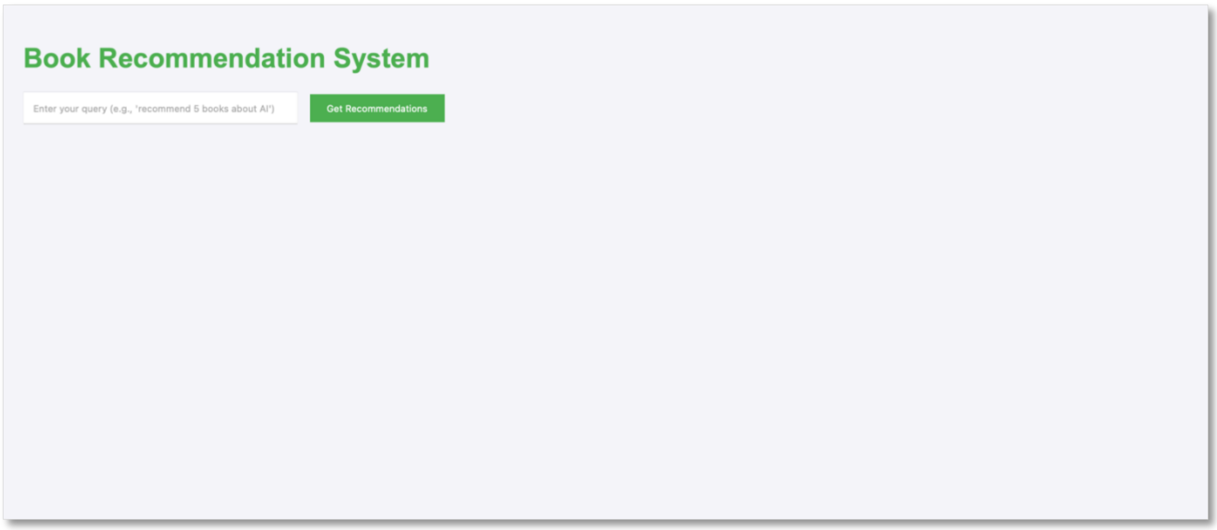
The Flask Agent for Book Recommendations focuses on being simple and easy to use, allowing users to navigate the application without any hassle. The HTML user interface is clean and straightforward, making it easy for users to type in their queries and receive book recommendations that fit their interests.

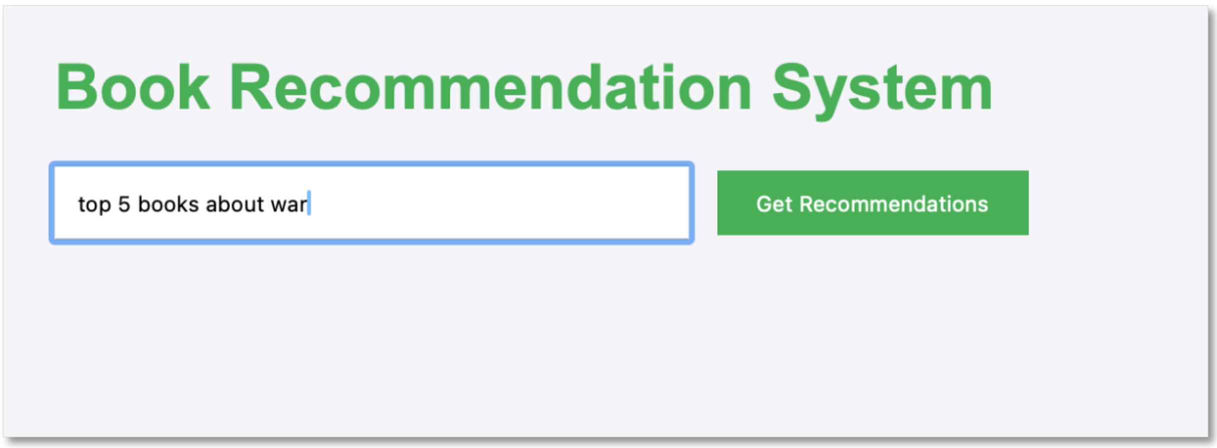
Users can easily input keywords or phrases related to what they like to read, such as “love” or “books about love.” This simple way of entering information encourages them to explore different literary themes. Once the user submits their query, the application processes it and shows a list of recommended books. Each suggestion includes important details like the title, author, and year of publication, helping users quickly find their next read.

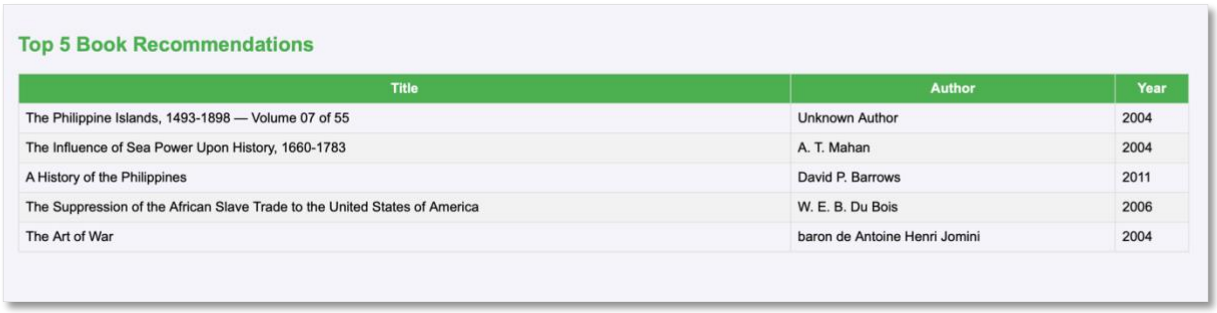
The web interface is designed thoughtfully using Flask, including clear prompts and a simple layout creating a user-friendly environment. When a user submits a query, it goes through the same natural language processing (NLP) system used for the book descriptions. This means the input is formatted in a way that can be easily compared with the books in the dataset. Based on this analysis, the application provides the top recommendations, making sure the results are relevant.

In the case that the user does not specify the number of books in the unput query, the application automatically present ten suggestions by default. However, if the input mentions a specific number in the query, like “give me top 5 books about love,” the system recognizes this and adjusts the output to deliver the requested number of results. This combination of a user-friendly interface, efficient query handling, and a robust recommendation algorithm creates a powerful tool for book discovery, enhancing the overall user experience and fulfilling the project’s goal of facilitating literature accessibility.

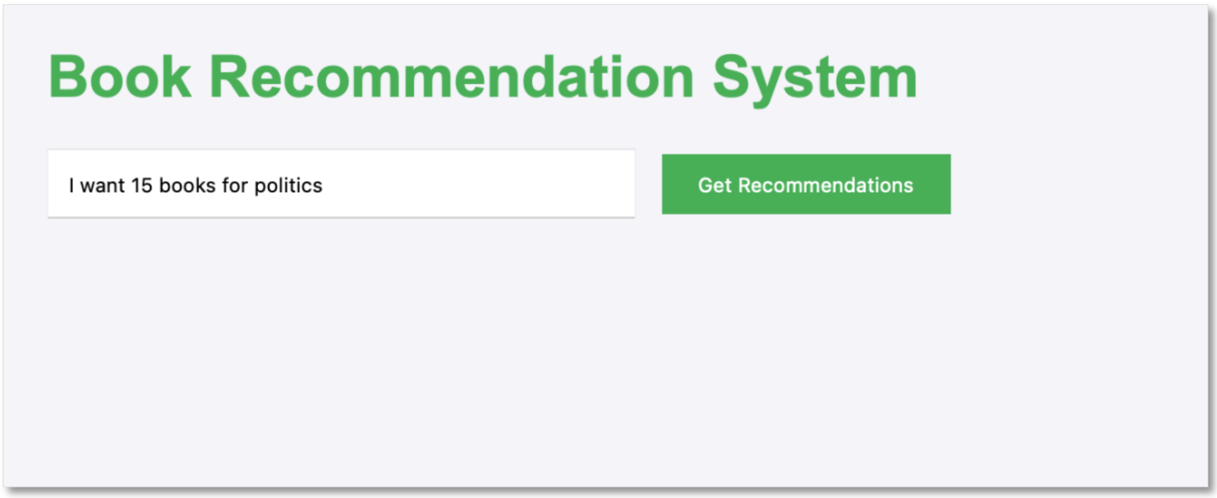
An overview of the interface of the agent is presented with print screens as follows:


Book Recommendation System
Enter your query (e.g., 'recommend 5 books about AI')
Get Recommendations



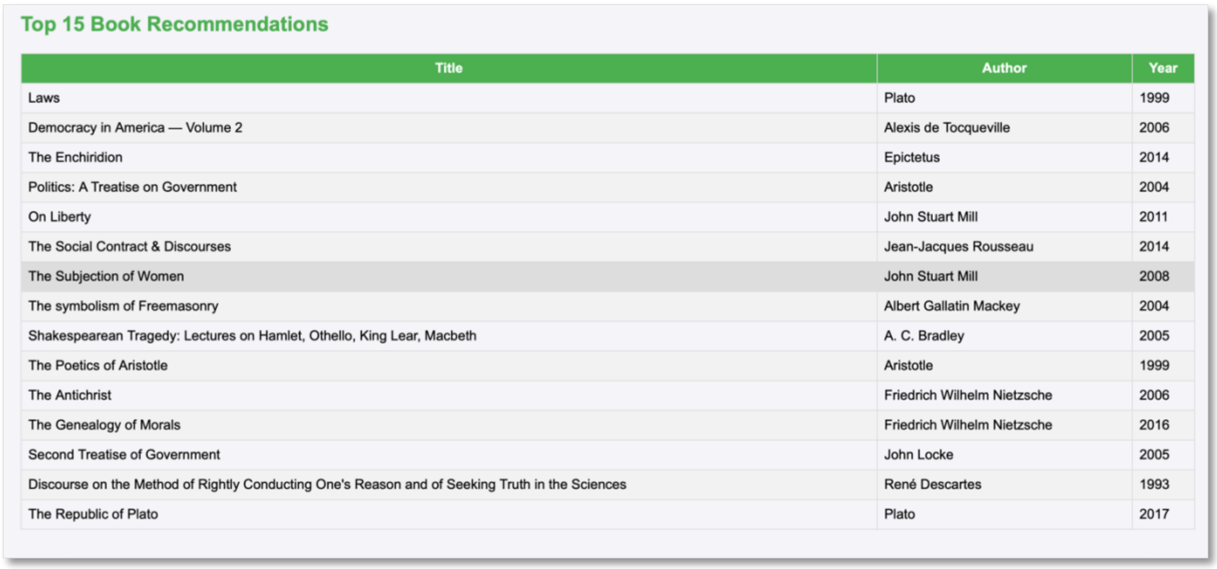

Top 5 Book Recommendations

Title	Author	Year
The Philippine Islands, 1493-1898 — Volume 07 of 55	Unknown Author	2004
The Influence of Sea Power Upon History, 1660-1783	A. T. Mahan	2004
A History of the Philippines	David P. Barrows	2011
The Suppression of the African Slave Trade to the United States of America	W. E. B. Du Bois	2006
The Art of War	baron de Antoine Henri Jomini	2004



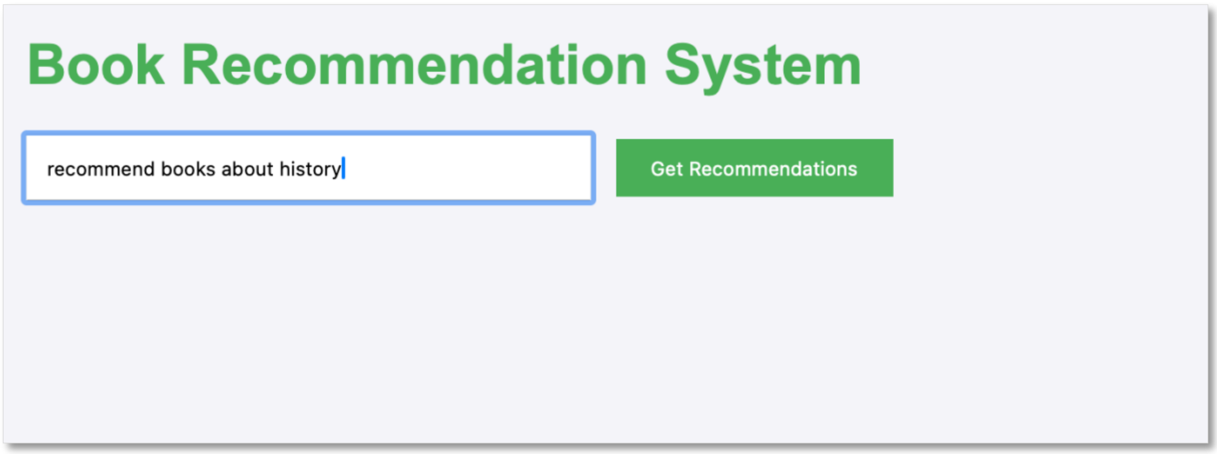
Book Recommendation System

I want 15 books for politics



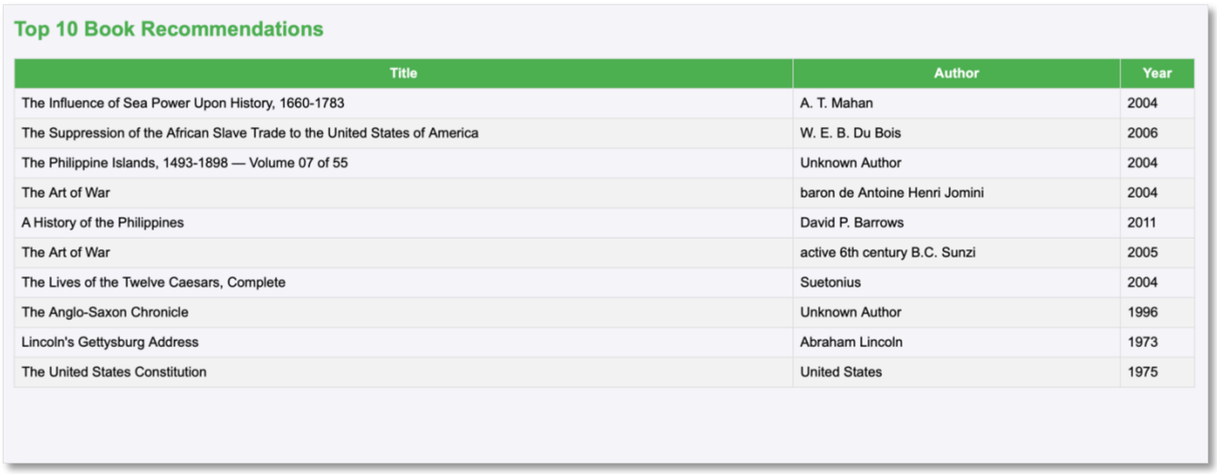
Top 15 Book Recommendations

Title	Author	Year
Laws	Plato	1999
Democracy in America — Volume 2	Alexis de Tocqueville	2006
The Enchiridion	Epictetus	2014
Politics: A Treatise on Government	Aristotle	2004
On Liberty	John Stuart Mill	2011
The Social Contract & Discourses	Jean-Jacques Rousseau	2014
The Subjection of Women	John Stuart Mill	2008
The symbolism of Freemasonry	Albert Gallatin Mackey	2004
Shakespearean Tragedy: Lectures on Hamlet, Othello, King Lear, Macbeth	A. C. Bradley	2005
The Poetics of Aristotle	Aristotle	1999
The Antichrist	Friedrich Wilhelm Nietzsche	2006
The Genealogy of Morals	Friedrich Wilhelm Nietzsche	2016
Second Treatise of Government	John Locke	2005
Discourse on the Method of Rightly Conducting One's Reason and of Seeking Truth in the Sciences	René Descartes	1993
The Republic of Plato	Plato	2017



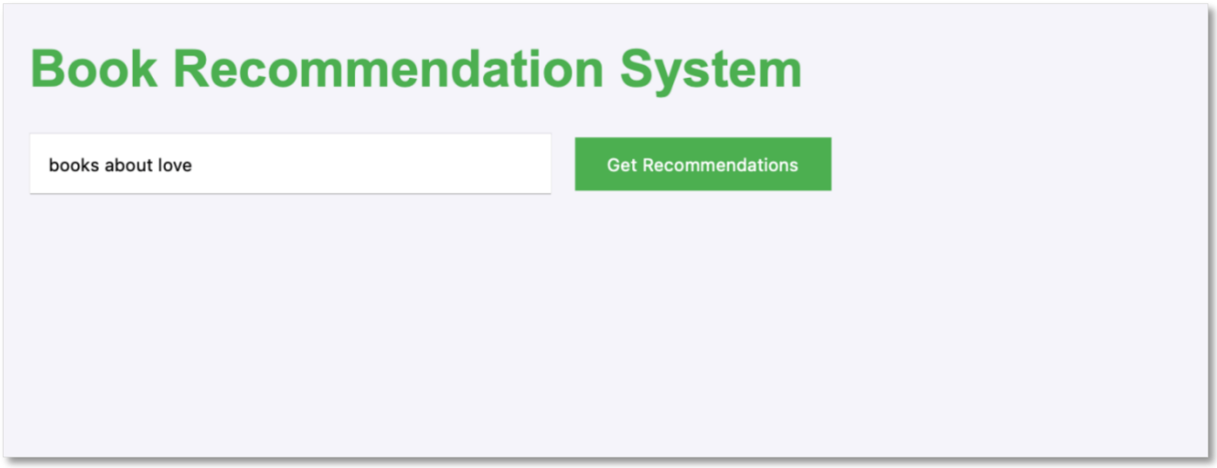
Book Recommendation System

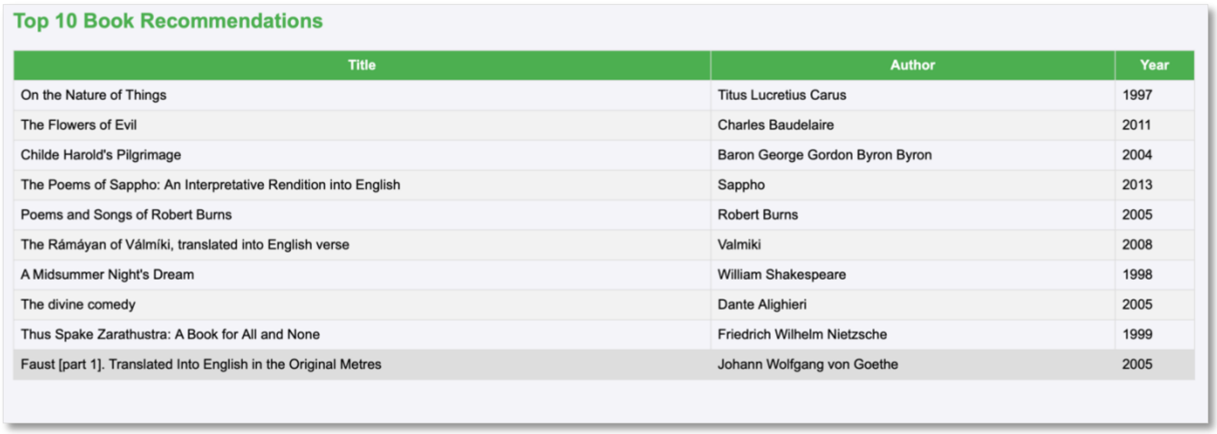
recommend books about history



Top 10 Book Recommendations

Title	Author	Year
The Influence of Sea Power Upon History, 1660-1783	A. T. Mahan	2004
The Suppression of the African Slave Trade to the United States of America	W. E. B. Du Bois	2006
The Philippine Islands, 1493-1898 — Volume 07 of 55	Unknown Author	2004
The Art of War	baron de Antoine Henri Jomini	2004
A History of the Philippines	David P. Barrows	2011
The Art of War	active 6th century B.C. Sunzi	2005
The Lives of the Twelve Caesars, Complete	Suetonius	2004
The Anglo-Saxon Chronicle	Unknown Author	1996
Lincoln's Gettysburg Address	Abraham Lincoln	1973
The United States Constitution	United States	1975





Top 10 Book Recommendations

Title	Author	Year
On the Nature of Things	Titus Lucretius Carus	1997
The Flowers of Evil	Charles Baudelaire	2011
Childe Harold's Pilgrimage	Baron George Gordon Byron Byron	2004
The Poems of Sappho: An Interpretative Rendition into English	Sappho	2013
Poems and Songs of Robert Burns	Robert Burns	2005
The Rámáyan of Válmíki, translated into English verse	Valmiki	2008
A Midsummer Night's Dream	William Shakespeare	1998
The divine comedy	Dante Alighieri	2005
Thus Spake Zarathustra: A Book for All and None	Friedrich Wilhelm Nietzsche	1999
Faust [part 1]. Translated Into English in the Original Metres	Johann Wolfgang von Goethe	2005

By taking a structured approach, the Flask Agent for Book Recommendations offers users helpful and relevant book suggestions while ensuring that the experience remains smooth and straightforward. This application brings together natural language processing techniques, machine learning models, and an easy-to-use web interface, making it a great tool for finding new books and supporting the main goal of making literature more accessible.

That being said, the volume of our input data did present some difficulties for us. Since the largest CSV file we could handle was used, some categories—such as specialized subjects like "psychotherapy"—may not be well-represented or might not exist at all. This means that if someone searches for those particular themes, they might not find any recommendations because not every genre is included in the dataset.

Not to be overlooked is the fact that this application operates locally. This option minimizes wait times for user enquiries and enables faster processing times. Because distributing huge datasets online might present a number of issues, we decided to employ a local arrangement. Large data hosting can lead to many problems and performance problems, particularly when using online services with tight capacity restrictions.

Remaining local allows customers to keep more control over their data and benefit from speedier answers. Additionally, it makes development and debugging easier without the typical limitations associated with cloud systems. In conclusion, one example of how web technologies might improve book discovery is the Flask Agent for Book Recommendations. It is an efficient tool for readers looking for books providing information about the title the author and the year of publish.

6 Challenges

There were many obstacles in the project that we faced. One of the first problems arose when we had to scrape the dataset from Project Gutenberg. There was no way to directly download collections of books, and we would have to scrape the site. And this meant building a scraping script that would obtain full texts of hundreds of books as fast as possible, manage differences in formatting between titles and guarantee we were retrieving books and their full texts. Fine-tuning the model was also tricky, particularly in choosing the right number of features and topics for the NMF model. We needed to balance it, ensuring the topics were meaningful but not too complex taking into consideration the size of our dataset. It took several iterations of testing different parameters before we settled on our model.

Issues encountered in implementing the Flask Project Locally

- Data Size Limitations: The biggest challenge was handling the enormous dataset that will be used in the book recommendation system. But as with most free hosted services, we were restricted in the high volume of data that could be deployed live online. Which meant we had to reduce (significantly) the size of our full dataset if we wanted to put our project into production. Since the solution stores books as files, it was not possible add them all to the CSV input file (due to storage limits of some platforms). Because of this, there might not be as many relevant recommendations for some categories, such as the "psychotherapy" category.
- File Organization: This code had multiple files from Python scripts to HTML templates and pre-trained models. Making sense of that chaos was interesting especially since it means making the connections between all these things, so they work well. One thing was more difficult when using Flask with its specific structure with templates, static files and Python code in different folders — it was hard to keep from disorder everything and place right file path.

- Limited Access: Since the flask project was implemented locally, which is not available to anyone who has a web link, demanded that users download the code, create a virtual environment, install dependencies and run the program themselves.

7 Conclusion

To sum up, this assignment managed to create a personalized book recommendation system using machine learning, and NLP techniques, with topic analyzed using NMF. The data were collected through web scraping from Project Gutenberg and they consisted of 930 books in text format. Key metadata suitable for the analysis were extracted and stored in a csv file which consisted of 930 rows, each on one for a single book. Most of the books are in English, and the majority of them contain around 200,000 – 600,000 characters. An additional word frequency analysis was also conducted during the data analysis which suggested a variety of topics like history, adventure, romance, politics, giving an initial insight of the topics identified NMF analysis.

Taking into consideration the above analysis, topic modeling can prove a useful way of categorizing large pieces of text (such as books but not limited to this) since this method clusters the words contained in documents into topics based on frequency. This bypasses the limitation of summarization techniques which are unable to drill down to the main ideas or themes of a long text.

However, this approach requires substantial text preprocessing in order to prepare the data inputs into readily processable chunks but also in order to ensure uniformity and consistency in their type and format. This can be achieved through publicly available libraries in python which achieve the removal of stopwords, lowercasing and vectorization through the TF-IDF process.

Topic Modelling, in terms of applicability in real life business scenarios, could become very useful in the following scenarios. One such case could be the implementation of this recommendation system as tool used by bookshop employees in order to suggest relevant book recommendation to interested customers. At this point, we should mentioned that this solution could be further enhanced by employing additional functionalities such as filters by publication year, author and/or publication agency in order to achieve more accurate and relevant results. Additionally, this recommendation system could prove useful in completely different scenarios such as this of academic paper classification. The continuously expanding database of available academic publications requires a way of categorizing them into topics while on the other hand academics could also be interested in finding relevant publications from past years.

More specifically, in this project the part of the publicly available database of books of Gutenberg was used in order to classify them into topics and provide relevant recommendations. After having processed the books downloaded as described section 2.2 the NMF procedure was implemented to extract the topics present in our dataset.

Finally, after having extracted the topics of each document, in the final stage of the project, the agent recommending books was implemented, which allows the user by entering a word or/and a sentence to display a list of recommended books, authors and year of publication of the books. The flask project designed to be a user-friendly option providing easy access to those interested in finding books. As mentioned in the section 6 Challenges, the interface was implemented locally due to limitations in the volume of data, leaving the possibility for further potential optimizations of the work in the future.