In the name of god

Personalized Book Recommendation System with NLP

Producer:Mohammad Kianinasab

Other members of group:

Mohammad Reza Majed

Masih Tanoorsaz

Sara Chatraee

Overview on NLP Techniques for Content-Based Recommender Systems for Books

Melania Berbatova

Sofia University ”St. Kliment Ohridski”

melania.berbatova@uni-sofia.bg

Overview on NLP Techniques for Content-Based Recommender Systems for Books

Melania Berbatova
Sofia University "St. Kliment Ohridski"
melania.berbatova@uni-sofia.bg

Abstract

Recommender systems are an essential part of today's largest websites. Without them, it would be hard for users to find the right products and content. One of the most popular methods for recommendations is content-based filtering. It relies on analysing product metadata, a great part of which is textual data. Despite their frequent use, there is still no standard procedure for developing and evaluating content-based recommenders. In this paper, we first examine current approaches for designing, training and evaluating recommender systems based on textual data for books recommendations for the GoodReads website. We examine critically existing methods and suggest how natural language techniques could be employed for the improvement of content-based recommenders.

Nomenclature

CBF	Content-based filtering
CF	Collaborative filtering
RS	Recommender systems

1 Introduction

Recommendation systems are engines that use algorithms leveraging the interaction between users to generate personalized recommendations. They provide users with recommendations for new content these users might be interested in (music, movies, books, etc).

Recommendation systems can be divided into three main types: Collaborative Filtering (CF), Content-based Filtering (CBF) and Hybrid systems. Collaborative filtering systems analyze users interactions with the items (e.g. through ratings, likes or clicks) to create recommendations. On the other hand, content-based systems use semantic information (frequently called metadata) about the items in the system. Hybrid systems are a combination of these two approaches. If compared to collaborative or content-based systems, hybrid ones usually exhibit higher recommendation accuracy. This is due to the fact that CF lacks information about domain dependencies, while CBF systems do not take into account users preferences.(Krasnoshchok and Lamo, 2014)

Collaborative filtering recommenders are systems that suggest recommendations based on users interactions (most commonly, ratings). A great deal of the most efficient collaborative filtering algorithms are based on the matrix factorization (MF). Matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices. (Koren et al., 2009) This family of methods gained popularity around the Netflix prize challenge and showed state-of-art results for many RS tasks. However, in this paper we will focus on content-based methods, as they can benefit from natural language techniques and increase the accuracy of recommendations in a hybrid approach.

Content-based recommenders are a type of recommender systems that use item metadata (description, rating, products features, reviews, tags, genres) to find items, similar to those the user has enjoyed in the past. To generate recommendations, we use items that are most similar to the ones liked by a given user. In the context of books, main characteristics, and even whole book content can be present as metadata. As descriptions and reviews are purely natural language data, and categorical data such as tags and genres can also be represented in a way suitable for natural language processing, employing such techniques is crucial for the design of successful content-based recom-

Proceedings of the Student Research Workshop associated with RANLP-2019, pages 55–61,
Varna, Bulgaria, Sep 2–4, 2019.
https://doi.org/10.26615/issn.2603-2821.2019_009

Abstract

Recommender systems are an essential part of today's largest websites . Without them , it would be hard for users to find the right products and content.

One of the most popular methods for recommendations is content-based filtering . It relies on analyzing product metadata, a great part of which is textual data. Despite their frequent use, there is still no standard procedure for developing and evaluating content-based recommenders .

In this paper, we first examine current approaches for designing, training and evaluating recommender systems based on textual data for books recommendations for the GoodReads website. We examine critically existing methods and suggest how natural language techniques could be employed for the improvement of content-based recommenders.

1-Introduction

Recommendation systems are engines that use algorithms leveraging the interaction between users to generate personalized recommendations. They provide users with recommendations for new content these users might be interested in (music,movies, books, etc).

Recommendation systems can be divided into three main types:

- Collaborative Filtering (CF)
- Content-based Filtering (CBF)
- Hybrid Systems

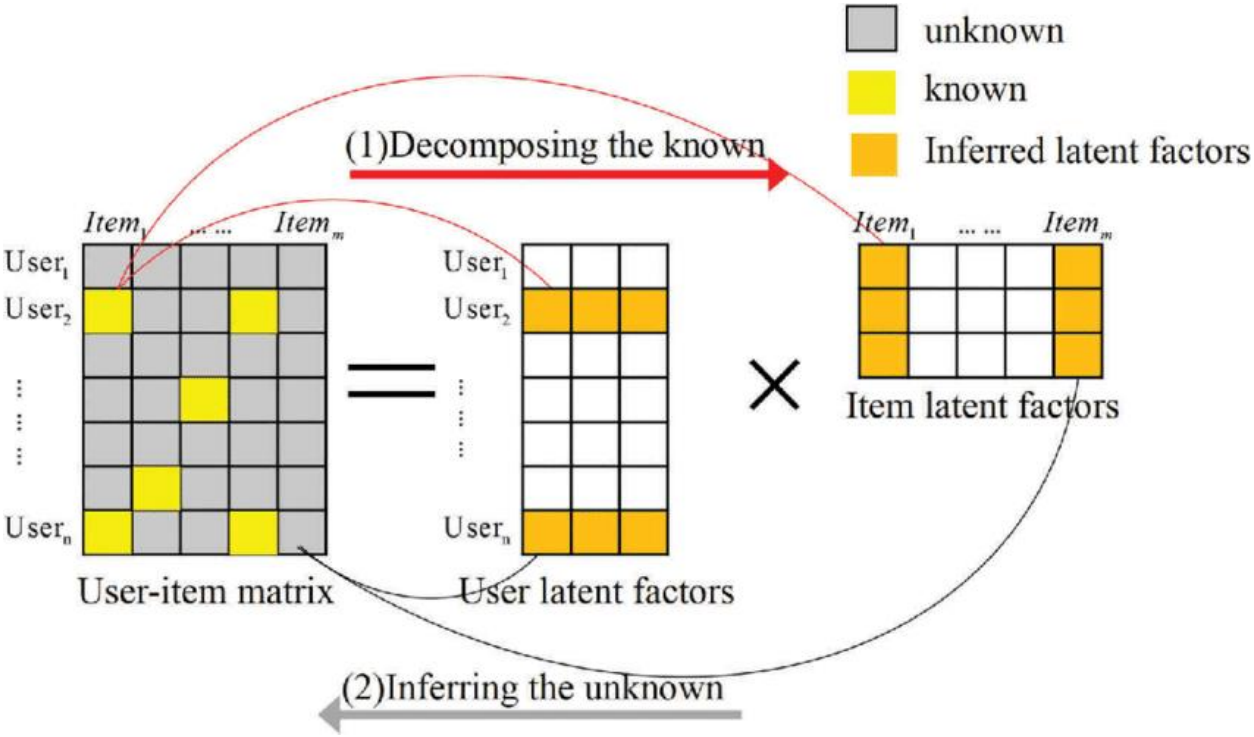
Collaborative filtering systems analyze users interactions with the items (e.g. through rat-

ings, likes or clicks) to create recommendations.

content-based systems use semantic information (frequently called metadata) about the items in the system.

Hybrid systems are a combination of these two approaches.

Collaborative filtering recommenders suggest recommendations based on user interactions, typically ratings. Many efficient algorithms in this field rely on matrix factorization (MF). These algorithms decompose the user-item interaction matrix into two lower-dimensional rectangular matrices (Koren et al., 2009). However, this paper focuses on content-based methods, which leverage natural language techniques to enhance recommendation accuracy in a hybrid approach.



Content-based recommenders use item metadata (such as descriptions, ratings, product features, reviews, tags, and genres) to suggest items similar to those a user has previously enjoyed. In the case of books, metadata can include detailed characteristics and even the content of the book itself.

2-Dataset

The Goodbooks-10k dataset contains 5,976,479 ratings for the top 10,000 books on Goodreads, alongside metadata for each book. It's accessible on the FastML website and organized into 5 files comprising ratings, book metadata, to-read tags, and user tags and shelves. User ratings are distributed with an average of 100 ratings per user, where the mean rating per user is 4. Compared to datasets like Book-Crossing, LitRec, and LibraryThing, Goodbooks-10k excels in volume, approaching 6 million records, which supports experimentation with various algorithms, particularly those designed for big data analytics.

3 Related Work

In their paper "A survey of book recommender systems" (Alharthi et al., 2017), a comprehensive overview of various approaches to book recommendation is presented, based on a compilation of over 30 papers up to 2017. These studies primarily utilize datasets such as Book-Crossing, LitRec, LibraryThing, INEX, and Amazon reviews, reporting results from methods including Content-Based Filtering (CBF), Collaborative Filtering (CF), and others. Among these datasets, only the LitRec dataset (Vaz et al.) incorporates data from GoodReads. In our current study, the focus is on models developed for GoodReads using the goodbooks-10k dataset. Recent publications predominantly concentrate on collaborative filtering techniques for this dataset. Among the 11 unique English papers retrieved from Google Scholar on recommender systems using the goodbooks-10k dataset (Le, 2019; Kula, 2017; Recommendation; Greenquist et al., 2019; Zhang et al., 2019, 2018; Paudel et al., 2018; Khanom et al., 2019; Kouris et al., 2018; Yang et al., 2018; Hiranandani et al., 2019), 10 explore collaborative filtering algorithms, two (Le, 2019; Greenquist et al., 2019) implement hybrid systems, and only one (Le, 2019) implements a straightforward content-based recommender. This study will specifically examine the content-based systems or components of hybrid systems developed using the goodbooks-10k dataset and compare them with systems utilizing another dataset for GoodReads, such as LitRec.

3.1-Overview

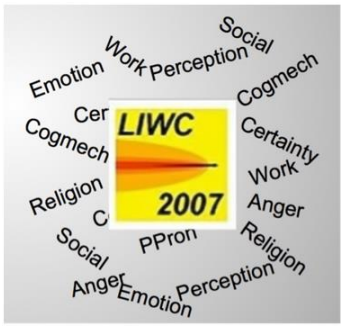
An overview of published content-based approaches for GoodReads is shown in Table 1.

Authors	Dataset	Features	Evaluation metrics	Algorithms	Dataset creation
Le (2019)	goodbooks-10k	ratings	MAP, CC, MPS, MNS, MDS	cosine similarity	test set - of 5-star ratings
Greenquist et al. (2019)	goodbooks-10k	tf-idf vectors	RMSE	cosine similarity	5+ ratings per user
(Alharthi and Inkpen, 2019)	Litrec	linguistic and stylometry features	precision@10, recall@10	kNN	10+ rating per user

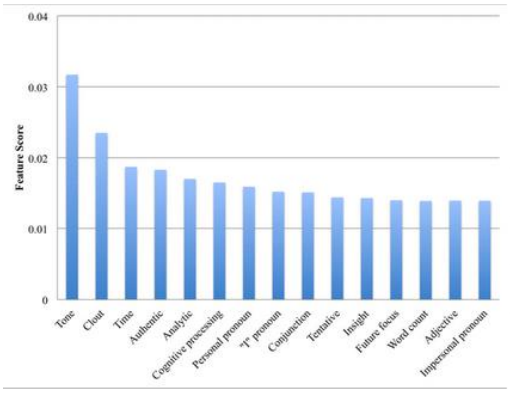
Table 1: Overview of recent published papers

- In their bachelor thesis, Le (2019) implement simple collaborative, content-based and hybrid systems for book recommendations . Their content-based recommender uses only ratings data and leaves aside books metadata. They achieve a best score of 0.842 of root-mean-square error(RMSE) for FunkSVD algorithm.
- Greenquist et al. (2019) implement a CBF/CF hybrid system. To gather more information, they merge goodbooks-10k data with Amazon reviews data . For books representations, they use tf-idf vectors of the books descriptions, tags, and shelves. Authors report using book descriptions in their content-based approach, but it is unclear how they obtained the descriptions, as the latter are not present in the goodbooks-10k dataset.
- Alharthi and Inkpen (2019) use the Litrec dataset to develop a book recommendation system based on the linguistic features of the books.Litrec dataset has ratings of 1,927 users of 3,710 literary books and contains the complete text of books tagged with part-of-speech labels.

For linguistics analysis, the authors use Linguistic Inquiry and Word Count (LIWC) (Tausczik andPennebaker, 2010), which is a popular resource that focuses on grammatical, content and psychological word categories. Using LIWC 2015 dictionary, Alharthi and Inkpen compute 94 categories,such as: percent of latinate words, function,words,affect words, social words, perpetual processes etc.



LIWC2007 logo represented with some word categories. Source: Author image



3.2 Critiques on Content-Based Recommenders

- Lack of usage of textual data such as tags available

Even in the case of using tags,no attention has been paid to the fact of hierarchy and synonyms in tags.

In order to deal with hierarchy, ambiguity and synonyms, some data normalization and natural language processing techniques could be adopted such as tf-idf vectorization.

- Lcak of the advantage of recent advances in machine learning algorithms, especially deep learning, despite the large volume of the data available.

3.3 General Critiques

- Lack of standardization in dataset preparation

Firstly, some redefine a ”like” as having a rating of at least 3 stars, while others don’t provide a clear definition of a like.

Secondly, some drop users having less than 5 ratings, others less than 10, and yet others

seem not to take out any users.

And lastly, since the initial dataset does not come with established training and test sets, the way different researchers have performed the train/test split seem to diverge.

- Difference in evaluation metrics used

Without suitable, unified metrics, it would be impossible to credibly prove that the use of NLP techniquees will significantly improve RS performance on the current task.

Since general dataset preparation and the choice of the evaluation metrics is a considerable topic,not central for this research proposal, we would like to leave it for future research.

4-Experiments

There are 5,976,479 ratings in our dataset.We chose the standard split of 80% percent of randomly sampled ratings for training data and 20% for test data, resulting in 4,781,183 train ratings and 1,195,296 test ratings.

As shown in Figure 1, the higher ratings significantly outnumber the lower ones. Therefore, we chose to define a like as a rating of 4 or more stars,instead of 3 or more.

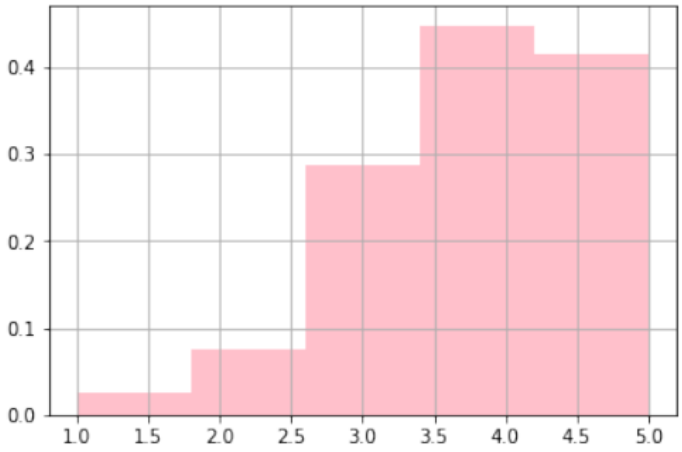


Figure 1: Distribution of ratings in the dataset

We used the python library TensorRec 3, which is a recommendation system library based on the deep learning library tensorflow, and employs the power of tensors and computational graphs. Recommender systems designed with TensorRec can be customized by choosing representation graphs,prediction graphs and loss graphs. Representation graphs are used for choosing the algorithms for latent representations (embeddings) of users an items. Prediction graphs are used for computing. Recommendation scores from the latent represen-

tations of users and items. Loss functions are the algorithms that define how loss is calculated from a set of recommendations.

We decided to work with recall@k as an evaluation metric. Recall@k shows the number of user's liked test items that were included in the top k of the predicted rankings, divided by k. Similar to Alharthi and Inkpen (2019), we chose k to be 10.We preferred recall@k than precision@k, because if the user has rated less than 10 books, precision@10 for their prediction cannot be 1. Finally,we preferred racall@k over RMSE and other metrics that measure the ratings prediction error, as for the design of the current recommender system, the exact score predicted is not as important as ranking the liked items higher than the not liked.

So far we have experimented with training simple CF and CBF systems. For our experiments, we used linear embeddings and weighted margin-rank batch (WMRB) (Liu and Natarajan, 2017) loss graph, which led to significantly better results on recall@10 than if optimizing for RMSE. For collaborative filtering we achieved 5.5% recall@10.

We utilized the year of publication and one-hot-encoded language as metadata features for our content-based approach. Our results showed a 0.98% recall@10. However, a major challenge we encountered was memory errors when attempting to train algorithms with larger feature sets, both locally and on popular cloud platforms like CodeLab and Kaggle. This limitation prevented us from fully evaluating the methods proposed in our current paper. Table 2 displays the scores achieved by both methods with varying parameters. As expected, approaches using 3-star ratings as 'like' showed poorer results. We expect to improve performance by several percentage points when we succeed in using more metadata features and we combine the designed CF and CBF in a hybrid system

Algorithm	Parameters	Recall@10
CF	Like = 3+ rating	4.69%
CF	Like = 4+ rating	5.52%
CBF	Like = 3+ rating	0.83%
CBF	Like = 4+ rating	0.98%

Table 2: Results

5 Suggested NLP Improvements

5.1 Using Tags Information

As we explained in the previous section, user tags information can be better utilized for the recommenders . The tags of a book show its genres (e.g.”youngadult”,”fiction”,”biography”), readers intents (”to-read”, ”to-buy”, ”to-be-finished”), books features (”printed”, ”books-in-spanish”),awards (”printz-award”), authors (”oscar-wilde),etc. and many of the tags have similar or equal meanings . Every book is characterized by its tags and the number of occurrences of every tag.

We propose using a bag-of-words word model of tags, as being more suitable than a tf-idf based one. After cleaning and bag-of-words vectorization, we can extract a variety or features. As Alharthi and Inkpen (2019) mention, we can use linguistic resources as Linguistic Inquiry and Word Count, or alternatively, predefined dictionaries of book genres to design features of books genres and vocabulary style.

Alternatively, we can use the bag-of-words representations of tags together with an unsupervised dimensionality reduction algorithm, as latent semantic analysis (LSA), to represent books.

Another approach is to use the power of word embeddings. Embeddings are used for transforming a word into a vector from a vector space with a fixed dimensionality, in a way that words occurring in similar contexts are represented by similar vectors. Current pre-trained word embeddings,such as word2vec (Mikolov et al., 2013), ELMo

(Peters et al., 2018) and BERT (Devlin et al.,2018) have proved to raise performance on many

natural language processing tasks, including text classification (Lai et al., 2015). We can use the weighted average of the tags tokens of a book as representation of the book.

5.2 Data Enrichment

In addition to a better exploitation of the available data, we can also gather new natural language data and extract additional features from it. As mentioned by Greenquist et al. (2019), it would be useful to work with book descriptions. We can easily scrape these descriptions from GoodReads website using the books IDs, or, if unavailable in certain cases, we can scrape them from Amazon.com book pages. From descriptions we can extract features as sentiment and distribution of adjectives,adverbs, nouns, and verbs; or we can represent descriptions as tf-idf vectors.

5.3 Alternative Approaches

An alternative algorithmic approach is to think of the recommendation task as a classification problem. For every user we can try to predict whether they ”like” or ”don’t like” certain set of books. In this setup, we can use books’ metadata together with classical approaches for text classification, as SVM or Naive Bayes, or incorporate deep learning algorithms such as recurrent neural networks (RNNs) and long-short

term memory (LSTM). If final predictions come with score for the labels given, we can sort the books in descending order of the scores for ”like”, take top 10 books in the sorted list and again measure recall@k. The downside of this approach is that it would not work well for users with too few ratings,as there will not be enough training data.

6 Conclusion

Many natural language processing techniques , such as extracting lexical, syntactic and stylometric features or word embeddings, can be used in content-based filtering for the recommendation of books. CBF systems employing these techniques can be used separately or in a hybrid with collaborative filtering. However, these techniques should be accompanied with standardized methods for dataset creation and result evaluation, so the results obtained are comparable to those from similar research.

Enhancing Personalized Recommendations: A Spacy-Based Hybrid Approach for Book Recommendation Systems

JAYANK TYAGI1, VANDANA CHOUDHARY2, NAMITA GOYAL3

1, 2, 3 Maharaja Agrasen Institute of Technology, Delhi, India

© JUN 2023 | IRE Journals | Volume 6 Issue 12 | ISSN: 2456-8880

Enhancing Personalized Recommendations: A Spacy-Based Hybrid Approach for Book Recommendation Systems

JAYANK TYAGI[1], VANDANA CHOUDHARY[2], NAMITA GOYAL[3]
[1, 2, 3] *Maharaja Agrasen Institute of Technology, Delhi, India*

Abstract- ***Recommendation systems have become ubiquitous in our digital age, with various platforms utilizing algorithms to predict what items users might be interested in and offer personalized recommendations. To build more effective recommendation models, it is essential to analyse and understand natural language, such as product descriptions or user reviews. Tokenization, part-of-speech tagging, named entity recognition, dependency parsing, and other features offered by Spacy, an open-source NLP toolkit, are effective tools that may be used to extract meaning from natural language text. Using Spacy, it is possible to identify the key topics or themes of a product description or review and match them to the interests of individual users. Spacy's speed and scalability make it well-suited for real-time recommendation systems that need to process large volumes of data quickly. As such, Spacy has become increasingly popular among researchers and practitioners working in this area, due to its unique set of NLP features and capabilities.***

I. INTRODUCTION

1.1 About Spacy

Recommendation systems have become ubiquitous in our digital age, with various platforms utilizing algorithms to predict what items users might be interested in and offer personalized recommendations. To build more effective recommendation models, it is essential to analyse and understand natural language, such as product descriptions or user reviews. Tokenization, part-of-speech tagging, named entity recognition, dependency parsing, and other features offered by Spacy, an open-source NLP toolkit, are effective tools that may be used to extract meaning from natural language text. Using Spacy, it is possible to identify the key topics or themes of a product description or review and match them to the interests of individual users. Spacy's speed and scalability make it well-suited for real-time recommendation systems that need to process large volumes of data quickly. As such, Spacy has become increasingly popular among researchers and practitioners working in this area, due to its unique set of NLP features and capabilities.

1.2 Types of recommendation systems

Recommendation systems can have various variants, including collaborative filtering, content-based filtering, and hybrid approaches that merge the two. Here's a brief overview of each type:

1. Collaborative filtering: based on the idea that people who have previously expressed preferences in a similar way are likely to do so in the future. Based on historical user behaviours, such as ratings or purchase histories, collaborative filtering algorithms produce tailored recommendations. The two most common types are user-based and item-based collaborative filtering. While user-based collaborative filtering makes suggestions based on the preferences of people who are similar to the user, item-based collaborative filtering makes suggestions that are similar to products the user has previously enjoyed [1].

2. Content-based filtering: This strategy suggests goods that are similar to those that a user has previously appreciated based on the properties of the things themselves. If a user likes a particular kind of music, an algorithm using content-based filtering can, for example, suggest other songs or performers. Content-based filtering may be especially useful when there is limited data available regarding user activities [2].

3. Hybrid approaches: These approaches combine content-based and cooperative filtering to benefit from each approach's strengths. Hybrid approaches can

Abstract

Recommendation systems are widely used in our digital era, with many platforms employing algorithms to predict user interests and provide personalized suggestions. To enhance the effectiveness of these systems, analyzing and understanding natural language is crucial, including parsing product descriptions or user reviews. Spacy, an open-source NLP toolkit, offers various features such as tokenization, part-of-speech tagging, named entity recognition, and dependency parsing. These tools enable extraction of meaning from text, helping to identify key topics or themes in product descriptions or reviews and matching them to individual user interests. Spacy's speed and scalability are advantageous for real-time recommendation systems that handle large data volumes swiftly. Consequently, Spacy has gained popularity among researchers and practitioners in this field due to its comprehensive NLP capabilities.

The growing importance of recommendation systems in digital platforms, powered significantly by Natural Language Processing (NLP) is very clear. Spacy, an open-source NLP library, is increasingly favored for its capabilities in analyzing and understanding natural language, essential for building effective recommendation models. Existing literature reviews reveal Spacy's successful applications in various recommendation systems, such as music, movies, and restaurants. It has been utilized for tasks like extracting entities from reviews and integrating sentiment analysis to enhance recommendation accuracy. However, research gaps include scalability comparisons with other NLP tools, exploration in niche domains like healthcare or finance, and addressing ethical concerns such as bias and privacy impacts. Future

studies aim to bridge these gaps and advance Spacy-based recommendation systems comprehensively.

IV. PROJECT DESCRIPTION

To get a better grasp and understanding of Spacy based recommendation systems,a project on book recommendation systems based on Spacy employ NLP and generate content-based recommendations was built . For the project data we used a data set scrapped by researchers at USCD as good reads did not have an API after 2016. The books metadata data set consisted of the metadata of 2.36 million books which contain data like the name, author, year of release, description , number of ratings etc. To generate accurate recommendations, we plan to employ several strategies like content based and collaborative filtering.

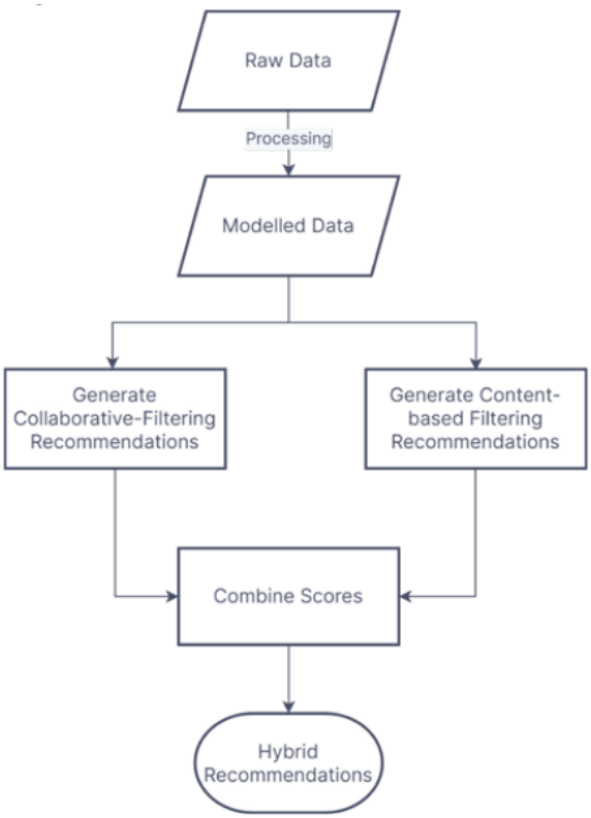


Figure 1: Steps to create a recommendation system

Before we could begin to even generate any recommendations, the data had to be modelled to isolate the fields that are essential to building a recommendation system. Firstly, To work with such a large dataset (~ 4GB), so a streaming technique of reading the data line by line to overcome memory limitations was employed. The dataset contained multiple versions of the same book so to overcome duplicity and generally improve the generated recommendations, only books with more than 15 ratings were considered for the dataset which returned 1.3 million books out of the 2.36 million.

```
books_titles = []
with gzip.open("goodreads_books.json.gz", 'r') as f:
    while True:
        line = f.readline()
        if not line:
            break
        fields = parse(line)

        try:
            ratings = int(fields["ratings"])
        except ValueError:
            continue
        if ratings > 15:
            books_titles.append(fields)
```

Figure 2: code filtering books with ratings > 15

To begin, a rudimentary search engine was needed to identify the book IDs of the books we wanted. The TF-IDF approach was used to reflect how important a word is to a document in a collection or corpus.

The term frequency-inverse document frequency (TF-IDF) is a statistical measure used to evaluate the importance of a word in a document or a corpus of documents. It is commonly employed in tasks such as information retrieval and natural language processing.

Term frequency (TF) represents the count of how often a specific word appears in a document. A higher term frequency suggests that the word is more significant within the document.

Inverse document frequency (IDF) calculates the frequency of a word's usage across all documents in a collection. Words that are rare or unique to a specific document have a higher IDF, whereas words that appear frequently across multiple documents have a lower IDF. The TF and IDF values are combined to compute the TF-IDF score for each word in a document. Words with higher TF-IDF scores are considered more important or relevant to the content of the document.

Now, to finally match our given search query with the given book titles cosine similarity was used. Cosine similarity is a measure used to compare two sequences of numbers by taking the cosine of the angle between them. These sequences are treated as vectors in a mathematical space, and the cosine similarity is determined by the dot product of the vectors divided by the product of their lengths. The magnitude of the vectors is not important in this calculation, only the angle between them. This metric is often used in data analysis to compare numerical sequences.

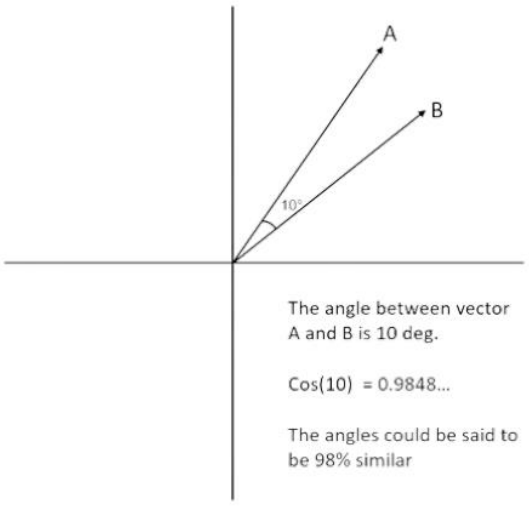


Figure 4: Cosine similarity visualized

VII.GENERATING RECOMMENDATIONS

To generate recommendations, we need to find the list of all the users that have read and reviewed the same books as us and rated them highly (> 4 stars).

To find such users we used the “goodreads_interactions.csv” that contains all of the reviews and ratings books by each user. We create a list of all books that users who like the same books as us like. We will use this list of books to generate the recommendations.

```
with open("goodreads_interactions.csv","r") as f:
    while True:
        line = f.readline()
        if not line:
            break
        user_id, csv_id, _,rating, _ = line.split(",")

        if user_id in overlap_users:
            continue
        try:
            rating = int(rating)
        except ValueError:
            continue
        book_id = csv_book_mapping[csv_id]
        if book_id in liked_books and rating >=4:
            overlap_users.add(user_id)
```

Figure 5: Code showing a list of similar books

In a collaborative filtering approach, we first identify a list of similar users and the books they enjoyed, and then, using the scipy and sklearn libraries in Python,we calculate the cosine similarity between a specific user and all other users. Finally, we identify the 15 users who are most similar to that user based on their ratings data.In order to generate book recommendations, the code then combines the ratings information for these comparable individuals.

The first step involves creating a sparse matrix named "ratings_mat_coo" from the "interactions" DataFrame using the coo_matrix function from scipy, as described in a previous answer. Subsequently, this "ratings_mat_coo" matrix is converted into a Compressed Sparse Row (CSR) format matrix named "ratings_mat" using the tocsr method.

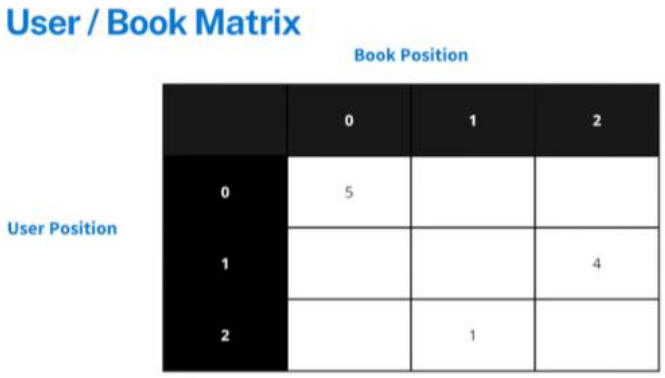


Figure 7: Illustrating a user/book matrix

The code utilizes the cosine_similarity function from the sklearn.metrics.pairwise module to calculate the cosine similarity between the ratings of a specific user (defined by the variable my_index) and all other users in the dataset. The resulting similarity matrix is then flattened into a 1D array using the flatten method.

Afterwards, the code employs the argpartition function from the numpy library to identify the indices of the 15 users who exhibit the highest similarity to the target user. These indices are stored in the variable indices

Next, the code filters the "interactions" DataFrame to include only interactions from users who are similar, using the "isin" method to select rows corresponding to user indices in the "indices" array. The filtered DataFrame is then stored in the "similar_users" variable.

Finally, using the "groupby" and "agg" methods, the code groups the "similar_users“ DataFrame by book ID and aggregates the rating data. The "count" and "mean" functions are used to get the number of ratings for each book and the average rating. The generated DataFrame, "book_recs," includes suggestions for the target user based on the ratings of other users.

To normalize ratings between 0 and 1 for effective hybrid recommendation generation, we employ the Min-Max normalization technique, also known as feature scaling. This method transforms numerical values in a dataset to a predefined range, typically 0 to 1. Here's how it works:

1. Identify the minimum and maximum values of the ratings dataset.

2. Subtract the minimum value from each rating and divide the result by the range (i.e., the difference between the maximum and minimum values).

3. This process scales the ratings linearly so that they fall within the range of 0 (smallest rating) to 1 (largest rating), preserving their relative positions.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

Figure 9: Min-Max Normalization

Collaborative filtering is a machine learning technique used in book recommendation systems. It analyzes past user behavior, such as purchased or reviewed books, to identify patterns. Based on these patterns, the algorithm suggests items that are similar to ones the user has shown interest in before.

The accuracy of a collaborative filtering-based book recommendation system is affected by the quality of the training data , modelcomplexity,and implementation-specific aspects

Of the recommendation algorithm. In general, collaborative filtering systems can be extremely good at providing suggestions, but there is always a trade-off between

recommendation accuracy and model computational complexity.

One way to enhance the accuracy of a collaborative filtering-based recommendation system is to incorporate additional information about the users and the items being recommended. For example, incorporating metadata about the books (e.g., genre,author, publication date) can help the system make more accurate recommendations by allowing it to take into account the characteristics of the items being recommended.

Overall, collaborative filtering is a potent approach for developing book recommendation systems, and when combined with the correct data and model design, it can yield very accurate suggestions.

A study published in the journal ACM Transactions on Information Systems found that amatrix factorization technique called singular value decomposition (SVD) achieved an accuracy rate of 80% on a dataset of movie ratings.

b. Content-based Recommendations

Before generating content-based recommendations, we needed to preprocess the book descriptions in our book metadata to create a book-vector graph using NLP. This allows us to easily find similar books based on cosine similarity between the user-liked book's generated vector and the book metadata book-vector.

First, we cleaned and preprocessed the data to ensure consistency and remove noise. This involved handling missing values, standardizing formats, removing duplicates, and performing other data cleaning techniques. We used SpaCy's built-in library to tokenize the book description, remove stop words, and convert everything to lowercase.

```python
nlp = spacy.load('en_core_web_lg')


def preprocess(desc):
    # Preprocess the book description using spaCy
    doc = nlp(desc)

    # Tokenize the description and remove stopwords and punctuation
    tokens = [token.lemma_.lower() for token in doc if not token.is_stop and not token.is_punct]

    # Join the tokens back together into a single string
    cleaned_desc = ' '.join(tokens)

    return cleaned_desc
```

Figure 11: Code for preprocessing book descriptions

The next step after preprocessing was to vectorize each book's description using an NLP model. This process generates numerical values that can be compared with vectors generated from descriptions of user-liked books. Vectorization converts textual data (book descriptions) into numerical representations suitable for machine learning algorithms. It allows us to measure similarity between book descriptions using mathematical operations. Representing descriptions as vectors enables calculation of distances or similarities to identify similar books.

```python
import concurrent.futures
import numpy as np

book_vectors = []

def process_description(desc):
    doc = nlp(desc)
    book_vector = doc.vector
    return book_vector
```

Figure 12: Code showing generated book-vector

Similarly, from our user-liked book data we can generate a vector of user liked books and compare the two vectors to find such similar books which the user may like with the help of cosine similarity. We aggregate the first 100k most similar books into our final recommendation dataframe.

```
# Generate the document vectors for the user's liked books
user_liked_vecs = np.array([nlp(desc).vector for desc in liked_books_df['clean_desc']])

# Calculate cosine similarity between user's liked books and all other books
similarities = cosine_similarity(user_liked_vecs, book_vectors)
```

Figure 13: Calculating cosine similarity between user-liked and book vectors.

We then input the cosine similarity scores in a new column ‘content_score’ to make combining scores with our earlier generated scores from collaborative filtering recommendations easier.

Spacy was used for this part of our system because of its comprehensive set of tools and functionalities for NLP tasks, making it a preferred choice for working with text data. Its ease of use, efficiency, language support,and integration with deep learning frameworks make it a valuable tool in the NLP ecosystem.

c. Combining scores to generate hybrid-recommendations

The recommendations are merged based on the book ID, and a combined score is calculated by adding the collaborative filtering score and the content-based score. The recommendations are then sorted based on the combined score in descending order, and the top n-recommendations books are selected as the final recommendations.

```
# Merge the collaborative filtering and content-based recommendations based on the book ID
merged_recs = collab_recs.merge(content_recs, on='book_id', how='inner')

# Calculate the combined score by adding the collaborative filtering score and the content-base
merged_recs['combined_score'] = merged_recs['normalized_score'] + merged_recs['content_score']

# Sort the recommendations by the combined score in descending order
merged_recs = merged_recs.sort_values('combined_score', ascending=False)

# Select the final recommended books with the highest combined scores
final_recommendations = merged_recs.head(10)
```

Figure 14: Merging two scores to generate combined score and provide recommendations.

Conclusion

This research investigates recommendation systems focusing on Spacy-based techniques. It introduces Spacy, a powerful NLP package, crucial in various NLP applications. The study explores collaborative filtering, content-based filtering, and hybrid approaches combining multiple strategies. The literature review underscores advancements in Spacy-based recommendation systems, highlighting their effectiveness in generating accurate, personalized suggestions. Additionally, it introduces a novel Spacy-based hybrid book recommendation engine that integrates collaborative filtering, content-based filtering, and Spacy's NLP capabilities to offer refined recommendations. In summary, this paper underscores how Spacy-based recommendation systems enhance personalized suggestions by integrating NLP with filtering approaches, potentially transforming recommendation systems by offering more accurate, contextually relevant ideas. Future research promises even more sophisticated Spacy-based recommendation engines as this field evolves.