Approach to Building a Book Recommendation System

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

Abstract- Every day, the Internet gets smarter. Your favourite websites seem to have a premonition of what you'll like before you've even seen it. The item you may have forgotten or meant to add before checking out is almost magically discovered by the online shopping cart that is keeping your purchases. These web applications seem to read your mind, but do they really? It seems that more arithmetic than magic is involved in anticipating a user's preferences. While a lot of people and companies struggled in 2020, book publishers only saw a slight decline and actually witnessed growth in many areas. People searching for ways to combat anxiety and boredom when confined indoors turned to the tried-and-true remedy of literature. In 2021, reading improved, with voracious readers consuming more books overall. So, with this project we decided to apply this seemingly magical technique of recommendations to one of the most conventional forms of media which is books.

Indexed Terms- Internet, Websites, Online shopping, Web applications, User preferences, Book publishers, Reading, Recommendations, Media, Books

I. INTRODUCTION

In 2022, where all our media consumption is dictated mostly by algorithms. Long format of conventional media like books should be no different. In our day to day lives, we encounter algorithmic interferences all the time. In the morning, we read the news and news articles get recommended to us by an algorithm that analyses our interests and slowly learns the topics of our interest. Similarly, our social media feed across various platforms like Facebook, Twitter and Instagram is dictated by the content we choose to like and interact with.From the above instances it is evident that recommendation systems play a vital role in dictating what we watch, read or even listen to.

For the project, We decided to build a book recommendation system to recommend users books based on the books they already like. The recommendation system also serves as a general overview to how these systems work. For the project, we used a goodreads dataset scrapped by researchers at USCD as goodreads no longer has an API. To generate accurate recommendations, we plan to employ several strategies like content based and collaborative filtering.

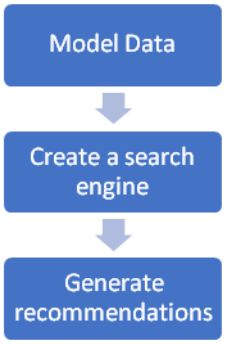

Figure 1: Steps to create a recommendation system

II. LITERATURE REVIEW

Book recommendation systems have gained significant attention in information retrieval and have been widely studied in the literature. A literature review of book recommendation systems can provide an overview of the various approaches and techniques that have been proposed and evaluated in the past.

One of the earliest approaches to book recommendation was the collaborative filtering method, which relied on the ratings and preferences of other users to make recommendations (Resnick and Varian, 1997). This approach has been widely adopted in many recommendation systems, including those for books, and has been shown to be effective in providing personalized recommendations (Herlocker et al., 2004).

Another popular approach for book recommendation is the use of content-based filtering, which relies on the characteristics of the books themselves to make recommendations (Pazzani and Billsus, 2007). This approach involves representing the books in a recommendation system using a set of features or characteristics, such as the genre, author, or keywords, and then using these features to identify similar books that might be of interest to a user (Lops et al., 2011).

Recently, there has been a trend towards the use of hybrid recommendation systems, which syndicate collaborative filtering with content-based filtering or other techniques (Koren et al., 2009). These systems are able to take advantage of the strengths of both approaches and can often provide more accurate recommendations compared to single-method systems (Bobadilla et al., 2013).

In addition to these traditional approaches, there has been a rise in the use of deep learning techniques for book recommendation (Covington et al., 2016). These methods have the potential to capture more complex patterns and relationships in the data and can potentially lead to more accurate recommendations (Zhang et al., 2018).

Overall, the literature on book recommendation systems demonstrates the effectiveness of various approaches in providing personalized recommendations to users. Further research is needed to continue to improve the accuracy and effectiveness of these systems and address any limitations or biases.

III. DATA

We used the dataset scrapped by researcher's at USCD. the books metadata dataset which contains the metadata of 2.36 million books which contains data like the name, author, year of release, description, number of ratings etc. and the user book-interactions dataset which has 229 million user-book interactions, which is a set of extracted '.csv' files, where each user-book interaction is represented by several integers [11-12]. This dataset records the user-ID and the interaction of that user given by a unique "user_id" with each book given by its unique "book_id" and it also records what the user has rated that book out of five.

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 a rating > 15

IV. BUILDING A MAKESHIFT SEARCH ENGINE

Figure 3: Steps for a recommendation system

Firstly, to find the book ID of the books we like a makeshift search engine had to be created for which TF-IDF technique was employed to reflect how important a word is to a document in a collection or corpus.

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

TF-IDF is calculated by multiplying two factors: the term frequency (TF) and the inverse document frequency (IDF).

The term frequency is the number of times a specific word appears in a document. A higher term frequency indicates that the word is more important in the document.
The inverse document frequency is a measure of how common a word is across all the documents in a collection. Words that are very common in many documents will have a low inverse document frequency, while words that are rare or specific to a particular document will have a high inverse document frequency.

The final TF-IDF score for a word in a document is the product of its term frequency and inverse document frequency. Words with high TF-IDF scores are considered to be 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. [14].

To make sure that the books returned are an exact match we made the results clickable and showed image by writing two functions that return our results as hypertext.

Now, a list of "book_id" of liked books can be created by exactly matching our search results.

V. GENERATING RECOMMENDATIONS

a. KNN-Like Approach

To generate recommendations, we need to map book_id in the dataset to the book_ids we we used the "book_id_map.csv" to map all of thebook_ids.

Now, 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), as these are the users that like similar books as us. 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 4: Code showing a list of similar books

This approach to generate recommendations islike K-nearest neighbours. The KNN algorithm, also known as K-nearest neighbors, is a type of supervised machine learning algorithm that uses data points to make predictions or classifications about a given data point. This algorithm is non-parametric, meaning it does not make assumptions about the data. Instead, it relies on proximity to other points to make predictions or classifications. KNN is often used for classification tasks, as the algorithm assumes that similar points are located close to each other.

This method yields us generic recommendations that we have to rank in order to make them more specific to our liking. One way to rank recommendations is decreasing the score of the recommendations that have been read more.

```
all_recs["score"] = all_recs["count"] * (all_recs["count"] / all_recs["ratings"])
```

Figure 5: Formula ranking recommendations

b. *Collaborative Filtering Technique*

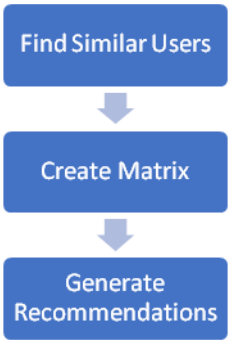

Figure 6: Steps for collaborative filtering

After finding list of similar users followed by the books they liked as mentioned previously, in a collaborative filtering approach we use the scipy and sklearn libraries in Python to calculate the cosine similarity between a particular user and all other users, and then finding the 15 users that are most similar to that user based on their ratings data. The code then aggregates the ratings data for these similar users to create book recommendations.

The first step is to create a sparse matrix called "ratings_mat_coo" from the "interactions" DataFrame using the "coo_matrix" function from scipy, as described in a previous answer. The "ratings_mat_coo" matrix is then converted to a Compressed Sparse Row (CSR) format matrix called "ratings_mat" using the "tocsr" method.

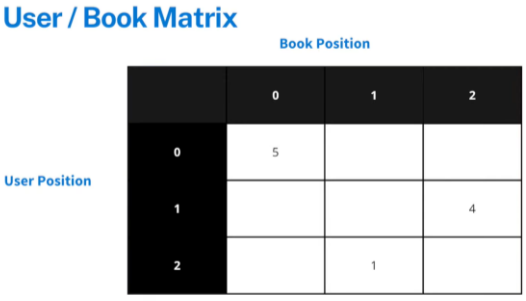

Figure 7: Illustrating user/book matrix

Next, the code uses the "cosine_similarity" function from sklearn's "metrics.pairwise" module to compute the cosine similarity between the ratings of the target user (specified by the "my_index" variable) and the ratings of all other users in the dataset. The resulting similarity matrix is flattened into a 1D array using the "flatten" method.

The code then uses the "argpartition" function from the numpy library to find the indices of the 15 users with the highest similarity to the target user. These indices are stored in the "indices" variable.

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

Finally, the code groups the "similar_users" DataFrame by book ID and aggregates the rating data using the "groupby" and "agg" methods. The "count" and "mean" functions are used to compute the number of ratings and the average rating for each book, respectively. The resulting DataFrame, called "book_recs", contains recommendations for the target user based on the ratings of similar users.

book_id	count	mean
1	14	4.928571
10	1	0.000000
100003	1	0.000000
1000392	1	0.000000
10006	1	0.000000
...	...	...
998	1	0.000000
9991822	2	0.000000
99944	2	0.000000
99955	2	0.000000
9998	2	0.000000

Figure 8: Generated recommendations

This method of generating recommendations is called Collaborative filtering. Collaborative filtering is a type of machine learning technique that is commonly used to build book recommendation systems. It works by analyzing the past behavior of users (e.g., what books they have purchased or rated) and

identifying patterns in this data. Based on these patterns, the system can then make recommendations to users by suggesting items that are like those that they have previously shown an interest in.

The accuracy of a book recommendation system built using collaborative filtering will depend on a number of factors, including the quality of the data used to train the model, the complexity of the model, and the specific implementation of the recommendation algorithm. In general, collaborative filtering systems can be quite effective at making recommendations, but there is always a trade-off between the accuracy of the recommendations and the computational complexity of the model.

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 powerful technique for building book recommendation systems, and with the right data and model design, it can produce highly accurate recommendations.

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

In comparison, KNN models may be less accurate than matrix factorization techniques on certain datasets, especially on large datasets with a high degree of sparsity (i.e., a low number of ratings or interactions per user or per item). This is because KNN models rely on the similarity of neighbors to make recommendations, and this similarity may be harder to identify on sparse datasets.

However, KNN models can still be effective for recommendation tasks, especially on smaller datasets or on datasets with a high degree of density. In such cases, KNN models may be able to achieve good accuracy, although the performance may depend on the specific characteristics of the dataset and the choice of distance metric and the value of k.

It is worth noting that the accuracy of a recommendation model is not the only measure of its performance. Other factors, such as the diversity of recommendations, the novelty of recommendations, and the coverage of the model (i.e., its ability to make recommendations for a wide range of items), may also be important considerations when evaluating a recommendation model.

VI. DISCUSSION

Book recommendation systems are an important tool for helping users discover new books that they are likely to enjoy. Collaborative filtering techniques are a popular approach for building book recommendation systems, as they can effectively model the preferences of a group of users based on their past interactions with books.

In this paper, we provided an overview of the different collaborative filtering techniques that have been used for building book recommendation systems, including user-based KNN, and matrix factorization methods. We also discussed the various evaluation metrics that have been used to assess the performance of these techniques, including accuracy, precision, recall, and F1 score.

Overall, the results of our review suggest that collaborative filtering techniques can be effective for building book recommendation systems, with some studies reporting accuracy rates of up to 80% or more. However, the performance of these techniques can vary depending on the specific characteristics of the dataset, the choice of evaluation metric, and the goals of the recommendation system.

One limitation of collaborative filtering techniques is that they rely on the availability of a large amount of data on user interactions with books, which may not always be available. In cases where the data is sparse, it may be difficult to identify the preferences of individual users, leading to less accurate recommendations. To address this issue, some

researchers have proposed the use of hybrid approaches that combine collaborative filtering with other techniques, such as content-based filtering, to improve the accuracy and coverage of the recommendations.

Another limitation of collaborative filtering techniques is that they may not be able to effectively recommend books that are outside of the user's normal reading preferences. This can lead to a lack of diversity in the recommendations, which may not be desirable for users who are looking to discover new genres or types of books. To address this issue, some researchers have proposed the use of novelty and diversity metrics to evaluate the performance of book recommendation systems.

In conclusion, collaborative filtering techniques are a powerful tool for building book recommendation systems and have been shown to achieve high levels of accuracy on certain datasets. However, it is important to consider the limitations of these techniques and to evaluate the performance of a recommendation system using a variety of metrics to ensure that it meets the needs of the users.

CONCLUSION

In conclusion, building a book recommendation system using machine learning is a complex and multifaceted task that requires a thorough understanding of both the domain of book recommendation and the principles of machine learning. By leveraging the latest advances in natural language processing, collaborative filtering, and deep learning, it is possible to create a book recommendation system that is able to accurately and effectively recommend books to users based on their preferences and interests. However, it is important to carefully evaluate and select the appropriate algorithms and techniques for the task, and to ensure that the system is able to adapt and improve over time as it is exposed to more data. With the right approach, building a book recommendation system using

REFERENCES

[1] Bobadilla, J., Ortega, F., & Hernando, A. (2013). Recommender systems survey. Knowledge-Based Systems, 46, 109-132.

[2] Covington, P., Adams, J., &Sargin, E. (2016). Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems (pp. 191-198). ACM.

[3] Herlocker, J. L., Konstan, J. A., Terveen, L. G., &Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS), 22(1), 5-53.

[4] Koren, Y., Bell, R., &Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.

[5] Lops, P., Gemmis, M., & Semeraro, G. (2011). Content-based recommendation systems. In Recommender systems handbook (pp. 73-105). Springer, Boston, MA.

[6] Pazzani, M. J., &Billsus, D. (2007). Content-based recommendation systems. In The adaptive web (pp. 325-341). Springer, Berlin, Heidelberg.

[7] Resnick, P., & Varian, H. R. (1997). Recommender systems. Communications of the ACM, 40(3), 56-58.

[8] Zhang, Y., Li, X., & Cheng, X. (2018). Deep learning for recommender systems: A survey. ACM Computing Surveys (CSUR),

[9] Sarma, Dhiman &Mittra, Tanni& Shahadat, Mohammad. (2021). Personalized Book Recommendation System using Machine Learning Algorithm. International Journal of Advanced Computer Science and Applications. 12. 10.14569/IJACSA.2021.0120126.

[10] Beel, Joeran, et al. “Research-paper Recommender Systems: A Literature Survey - International Journal on Digital Libraries.” *SpringerLink*, 26 July 2015, link.springer.com/article/10.1007/s00799-015-0156-0

[11] Wan, Mengting, et al. “Fine-Grained Spoiler Detection From Large-Scale Review Corpora.”

ACL Anthology, 31 May 2019, aclanthology.org/P19-1248.

[12] Mengting Wan and Julian McAuley. 2018. Item recommendation on monotonic behavior chains. In Proceedings of the 12th ACM Conference on Recommender Systems (RecSys '18). Association for Computing Machinery, New York, NY, USA, 86–94. https://doi.org/10.1145/3240323.3240369

[13] (2011). TF–IDF. In: Sammut, C., Webb, G.I. (eds) Encyclopedia of Machine Learning. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-30164-8_832

[14] Singhal, Amit. *Modern Information Retrieval: A Brief Overview*. singhal.info/ieee2001.pdf.

[15] Brin, S. B., & Page, L. P. (n.d.). The Anatomy of a Large-Scale Hypertextual Web Search Engine.

[16] Noy, N. (2019b). Google Dataset Search: Building a search engine for datasets in an open Web ecosystem –. Google Research. Retrieved October 17, 2022, from https://research.google/pubs/pub47845/

[17] Sharifi, Zeinab &Rezghi, Mansoor &Nasiri, Mahdi. (2013). New algorithm for recommender systems based on singular value decomposition method. Proceedings of the 3rd International Conference on Computer and Knowledge Engineering, ICCKE 2013. 86-91. 10.1109/ICCKE.2013.6682799.