Book Recommendation System

Abichal Ghosh (1225427294) Meesum Ali Khan (1225453632)
Nikhil Kumar Singh (1225658070) Soham Nag (1225463863)
Syed Asad Husain (1225380117)

December 8, 2022

Abstract

In this project, we explore different approaches that go into building a recommendation system and use book data to implement them. The two main approaches are called Content-Based and Collaborative Filtering, and the combination of these two is known as Hybrid Filtering. In Content-Based Filtering, we used different approaches such as TF-IDF and Word2Vec to vectorize textual data and find books with the highest similarity scores. In Collaborative Filtering, we used both memory-based and model-based algorithms to find books by grouping the users. We also implemented Hybrid Filtering by combining these two approaches, and the results for each approach are detailed in this paper.

1 Introduction and Motivation

The plethora of books available in the market today makes it difficult to manually discover new books to read. Good recommendation systems help in narrowing down our search using our reading history, likes/dislikes, and other features. We have decided to explore recommendation systems through books because we believe the features of book data make it complicated to make good recommendations and the challenge of solving this problem will give us a unique perspective. Book data is tricky because all the features we are dealing with are raw text-based, and it's hard to compare them quantitatively. Moreover, the results of our recommendations will be hard to objectively validate, as book preferences are very subjective.

As we will discuss in detail, there are three main approaches for a recommendation algorithm; Content-Based Filtering, which is based on the features of the item we are recommending, Collaborative Filtering, which is based on user data for the items, and Hybrid Filtering, which is a combination of the two. In the following sections, we will dive into the details of these three approaches, and analyze and compare the results.

Here, we implemented various machine learning models and approaches(Clustering, KNN). We performed statistical analysis in the form of similarity calculations, Singular Value Decomposition and many more. This enabled us to apply our learning from the Statistical Machine Learning (CSE 575) Course.

2 Problem Description

Book recommendation is inherently a tricky problem to solve due to the subjective nature of the preferences of individuals. This makes it incredibly hard to identify the features that are important to make recommendations. The currently available book datasets are lacking solely due to this reason thus the existing book recommendation systems are not able to capture the essence behind user-book relationships. Another major challenge is the evaluation of the results our models give. Since our models are based on unsupervised learning, we don't have a ground truth to evaluate the accuracy of our results.

3 Dataset

In this work the Content-Based and Collaborative filtering methods have been implemented based on 2 datasets. The Final hybrid model has been implemented on a coalition of the 2 datasets. The details of the datasets are as follows-

1. Book Crossing Dataset [1]

   The Book Crossing Dataset, contains ratings from 278,859 users on 271,379 books. This has been used for the collaborative filtering task.

2. CMU Book Summaries Dataset [2]

   The CMU Book Summaries Dataset has been used to extract the book summaries of 16,559 books along with aligned metadata (like author, title, genre, etc) from Freebase. This has been used for the content-based filtering task.

3. Merged Dataset

   Finally, the 2 datasets have been combined together for the Hybrid-Filtering Method.

4 Content-Based Filtering Model

In content-based filtering, we are using the various features of an item to make recommendations. In the case of our book dataset, we have access to attributes such as title, author, book summary, and genres. Using these attributes, we must find the similarity between one book and another. If we are able to achieve that, we can recommend books to a user based on the books they already like, by using this similarity measure.

4.1 Preprocessing

Since we are working with raw string data, preprocessing is a very important step. For all the text data that we have, we need to remove things like special characters, punctuations, stopwords, make everything lowercase, etc. This is mainly useful for our book summaries data, as future steps require us to find similar words and synonyms across summaries, and all words need to be in the same format for this to be possible. Our genre data was initially in a dictionary format, so we converted it into a single string with all genres for each book, and also added the author and title to that same string. We also clubbed multi-word genres into a single word. This would be useful in our future methods for comparing books.

4.2 Sample Input

We use a single input for all approaches and compare them on the basis of output recommendations received from each approach. Here, we took 'Moby Dick' as the input. The following book information about the input book will help us understand the comparison:

- Title: Moby-Dick; or, The Whale
- Author: Herman Melville
- Genres: seastory fiction adventurenovel
- Summary(one line): an observant young man setting out from Manhattan, has experience in the merchant marine but has recently decided his next voyage will be on a whaling ship.

4.3 Similarity Function

Recommendation systems use similarity functions to return the output that is most similar to the given input. Some of the most common and effective ways of calculating similarity are:

- Dot product:

$$Sim(A, B) = A.B = |A||B|Cos(\theta)$$

- Jaccard distance:

$$Sim(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

- Euclidean distance:

$$Sim(A, B) = |A - B|$$

- Cosine similarity:

$$Sim(A, B) = \frac{A.B}{|A||B|}$$

We use cosine similarity in content-based filtering because:

- Cosine similarity captures semantic similarity better than other above-mentioned criteria
- It is resilient to occurrence counts and heterogeneity of word neighborhood

4.4 Comparison Methods

There are three main methods that we have implemented, and each one is a progression on the previous one. The methods are:

- TF-IDF 4.5
- Average Word2Vec 4.6
- TF-IDF Word2Vec 4.7

We have used these methods on our book summary feature, and for the combined feature with title, author, and genres, we have used a simple CountVectorizer that will be defined later.

4.5 TF-IDF

TF-IDF stands for Term Frequency Inverse Document Frequency. Using this concept, we can find similar summaries based on the words that they share. One important aspect of TF-IDF is that it takes into consideration the weight of a word in a document. If a particular word is being repeated a lot in a document, that means it is important to that document. However, it also takes into account whether the word is a very common one, and if that is the case, it will reduce the weight that word has on the document. For two documents that are similar, we expect to see words that are relatively rare in other documents, but frequent in these two documents. This is the underlying principle of finding similarity using TF-IDF. The formula [3] for generating the word vector for a word is as follows:

$$w_{i,j} = tf_{i,j} \log(\frac{N}{df_i})$$

where,
$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

However, TF-IDF by itself is not a strong enough approach to give us good results by itself. We will use it later to enhance the other method that we use.

4.6 Average Word2Vec

The Word2Vec model is a shallow two-layer neural network with two variations. One variation predicts surrounding words based on a given word, and the other predicts a word given surrounding words. Here, we use Word2Vec to generate word embeddings. Through the process of training, the Word2Vec model generates word vectors for given words which, in the case of synonyms, are very similar to each other. This makes sense, as words that can be used interchangeably should have similar chances of being predicted in the same scenario.

In the TF-IDF approach, we had the disadvantage of not accounting for synonyms when we were comparing the similarity of documents. In the case of Word2Vec, synonyms are also captured. We represent summaries using the following vectorization formula[4]:

$$\frac{W2V(w_1) + W2V(w_2) + ... + W2V(w_{23})}{N}$$

On top of this, we use cosine similarity on the result to find the most similar summaries.

4.6.1 Analysis

Top 5 Recommendations:

```
Top 5:
The Phantom Freighter
Sampagitang Walang Bango
The Sea-Wolf
Pirates of Venus
The Rage of Hypsis
```

Figure 1: Recommendation using Average Word2Vec

Taking one recommendation from the list

- Title: Phantom Freighter
- Genres: mystery detectivefiction fiction

- Author: Franklin W.Dixon
- Summary: The Hardy brothers embark on a freighter trip under mysterious circumstances and find themselves involved with a smuggling ring.

As We can see from the top recommendation, the theme of sea and ships has been picked up by our Average Word2Vec implementation.

4.7 TF-IDF Word2Vec

To improve on our Average Word2Vec model, we add TF-IDF word vectors as well. We incorporate TF-IDF using the following formula[4]:

$$\frac{tf_1 W2V(w_1) + .. + tf_n W2V(w_n)}{tf_1 + tf_2 + tf_3 + ... + tf_n} \quad (1)$$

Where we are essentially weighting the Word2Vec Approach with the TF-IDF vector for each word in the document.

4.7.1 Analysis

Top 5 Recommendations:

```
Top 5:
Dick Sand, A Captain at Fifteen
Merlin Effect
The Poseidon Adventure
Island of the Blue Dolphins
Robinson Crusoe
```

Figure 2: Recommendation using TFIDF Word2Vec

Taking one recommendation from the list

- Title: Dick Sand, A Captain at Fifteen
- Genres: fiction adventurenovel
- Author: Jules Verne
- Summary: Dick Sand is a fifteen-year-old boy serving on the schooner "Pilgrim" as a sailor. The crew is whale hunters that voyage every year down to New Zealand.

Compared to our previous Average Word2Vec model, the TF-IDF enhanced model is spot on with its recommendation. The top recommendation picks up on the exact theme of sailor and whale hunting that is the basis of our input sample, "Moby Dick".

4.8 CountVectorizer

The TF-IDF Word2Vec approach is giving pretty strong results, but so far we have not used the other features like book genre and book author.

To incorporate these features, we have clubbed them all into a single string, and using a simple CountVectorizer, generated a word count vector for them. Now, the similarity would be based on whether there are identical words between different strings of different books. Whatever result we get for that, we assign a weight of 0.2 to it, While giving a weight of 0.8 to the summary similarity score. Hence, in situations where two books may be identical in summary similarity to our target book, genres, and the author can act as the difference.

4.8.1 Analysis

Top 5 Recommendations:

```
Top 5:
Dick Sand, A Captain at Fifteen
Robinson Crusoe
The Poseidon Adventure
The Old Man and the Sea
Island of the Blue Dolphins
```

Figure 3: TFIDF Word2Vec augmented by additional features

Taking one recommendation from the list

- Title: Old Man and the Sea
- Genres: seastory childrensliterature fiction novella
- Author: Ernest Hemingway
- Summary: The Old Man and the Sea is the story of a battle between an old, experienced Cuban fisherman and a large marlin.

Compared to our TF-IDF Word2Vec results, this one generates almost the same recommendations, with the replacement of "Old Man and the Sea". This replacement can be attributed to the use of genres, as the genres of this book are more similar to "Moby Dick" than "Robinson Crusoe", which is the book that was replaced. Thus, this gives

the best results, as it incorporates all the features of the dataset and gives meaningful recommendations based on them.

5 Collaborative Filtering Models

Collaborative Filtering Models are a type of Recommender System that extract information and generate recommendations based on matching/grouping users [5].

Collaborative Models can be mainly divided into 2 types namely memory-based Collaborative Filtering and model-based Collaborative Filtering [5]. Memory-based methods use the user database directly in order to make the predictions, while model-based methods use the user database to learn a model which is then used for predictions. In this work, a total of 3 different Models have been implemented, 1 memory-based model and 2 Model-based models.

1. Memory-Based Models
    - Simple Base Collaborative Filtering 5.1
2. Model-Based Models
    - KNN Based Collaborative Filtering 5.2
    - SVD Based Collaborative Filtering 5.3

The working and implementation of these 3 models is given in the following subsections.

5.1 Simple Base Memory Based Model

In this method, a simple memory-based Collaborative Filtering Model has been implemented. This model is an implementation of one of the methods presented in the paper [6].

In memory-based collaborative filtering models, the rating of a particular user on a given book is calculated based on the data of users and their ratings of books in the Training Dataset. The Training Dataset can hence be considered as the set of ratings r_{ij}, where r_{ij} corresponds to the rating of the user i on the book j.

The task of this memory-based model is to predict the rating of a user a on the book/item i, r_{ai}. This rating, r_{ai} is calculated by utilizing the information of the user a that is already known from the training data and some information of all other users on the book i. Thus, r_{ai} is calculated as the sum of the average rating of the user a and the weighted sum of the ratings all users in the training data on the book i. This is given by the equation 2

$$\begin{aligned} &r_{ai} = (\text{Average Rating of user a on all books}) + \\ &(\text{Weighted sum of ratings of users on the Book j} \\ &\implies r_{ai} = \bar{r}_a + \kappa \sum_{b=1}^{n} w(a,b)(r_{bi} - \bar{r}_b) \\ &\quad \text{, Where n is the number of users in Train} \end{aligned} \tag{2}$$

Here, the average rating of a user on books they have read in the Training Dataset, $\bar{r}_a$, is given by the following equation-

$$\begin{aligned} \bar{r}_a = \frac{1}{|I_a|} \sum_{i \in I_a} r_{ai} \\ \text{, } I_a \text{ is the set of books in the Dataset} \\ \text{which the user a has rated} \end{aligned} \tag{3}$$

Further, in equation 2, κ is a normalizing factor, and the weight calculation between the users, $w(a,b)$, is the Pearson Correlation Coefficient and is calculated as shown in equation 4

$$w(a,b) = \frac{\sum_i (r_{a,i} - \bar{r}_a)(r_{b,i} - \bar{r}_b)}{\sqrt{\sum_i (r_{a,i} - \bar{r}_a)^2 \sum_i (r_{b,i} - \bar{r}_b)^2}} \tag{4}$$

5.1.1 Analysis

Upon implementation, it is seen that this model gets the following results-

Metric	Whole Dataset	Part Dataset
RMSE	1.7215	1.9049
MAE	2.3033	2.7059

(5)

The model's RMSE values lie in a range of 1.7-2.3 which is much higher in comparison to state-of-the-art systems such as GLocal-K [7](RMSE - 0.89), and MG-GAT [8](RMSE - 0.89). This increase in the RMSE could be because of the Pearson Correlation Coefficient. Some better measures such as Cosine Similarity could be used here. Another reason is

due to the simplicity of the formula(equation 2). Hence more complex model-based methods have been used implemented in the following sections.

5.2 KNN Based Collaborative Filtering

K Nearest Neighbour is a machine learning algorithm that finds clusters of similar users based on common book ratings and make predictions using the average of top k nearest neighbors. KNN has the capability to perform both classification and regression tasks. It assumes that data points with similar features are closer to each other and form a cluster. It memorises the training data and simultaneously uses distance metric to calculate the relative similarity.

In this method-based collaborative filtering method, the KNN algorithm has been used as a model. Further, as the KNN algorithm is a clustering algorithm, this method can also be called as a cluster-based collaborative filtering method. Here, cosine similiarity(check 4.3 for formula) has been used as the measure for distance computation.

The data for any collaborative filtering is user-item matrix, where item(s) in our case are the books. From theoretical perspective, the data is not sparse in most of the cases. But from a practical perspective the data is very sparse, with up to 90%-95% missing values. If we think about it counter intuitively, if the matrix were not sparse, user would have rated most of the books, we wouldn't need a or any recommendation system.

To deal with the curse of dimensionality in our data, we have converted our user-item matrix in a compressed sparse rows (CSR) data structure.

CSR is nothing but a representation of sparse matrix using three - one dimensional arrays, that respectively contains non zero values, the extent of rows and columns indices.

5.2.1 Analysis

As this method falls under the category of unsupervised learning, there is no specific way to measure success. However, visually analyzing the books recommended, it was observed that KNN was not at par with the state-of-the-art latent-factor models. Hence, to improve the recommendations, SVD-Based Collaborative Filtering is explored in the next section.

5.3 SVD Based Collaborative Filtering

One of the most common approaches in collaborative filtering for recommendation systems is latent factor models. Latent factor models are state of the art methodology for model-based collaborative filtering. These models work with a basic assumption that there exists an unknown lower dimensional representation of users and items where user-item affinity could be modelled accurately [9]. One such popular model is the Singular Value Decomposition (SVD), which solves the problem of sparsity using feature extraction and correlation from user-item matrix.

SVD is a matrix factorization technique which reduces the data from NxM dimension with rank r information to Nxr relevant data.

We have reduced the size of our dataset by applying a couple of more filters on the existing dataset. For one, we have filtered out books with fewer than 10 ratings. Furthermore, we have kept books with non-zero ratings only. Lastly, we have dropped all the records where users had fewer than 11 interactions.

Matrix Factorization:

Matrix factorization is a simple embedding model. Many computers cannot solve complex matrix operations with any acceptable precission/accuracy. A matrix factorization model decomposed a complex and large matrix into two small factors. Intuitively, if there are 2000 users and

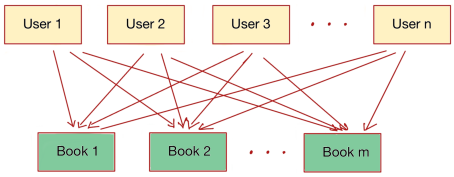


Figure 4: Before Matrix Factorization

1000 books, them there will be a total of 2,000,000 parameters. Now, if the factor number is kept as 100, then the number of features would be we will have 2000 x 100 features = 200,000 parameters , additionally, 100 x1000 = 100,000 parameters. If these 2 parameter values are added, then a total

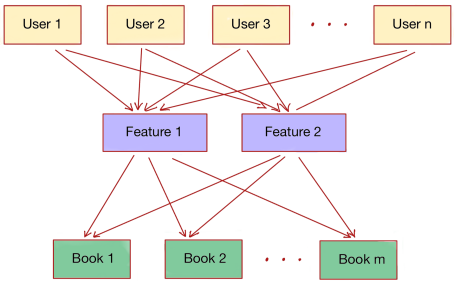


Figure 5: Matrix Factorization

of 300,000 parameters are achieved, which is much less in comparison to the 2 million parameters from the original matrix.

In the program, the factor has been given a value of 20.

5.3.1 Analysis

Implementing the SVD-Based model solved the problem of sparsity which was faced in the previous models. Thus this SVD-Based model is the best model implemented and outperforms both the previous Collaborative Filtering Models.

6 Hybrid Filtering Model

Having implemented and compared multiple methods for content-based and collaborative filtering models, a Hybrid Filtering Model is implemented using the best approaches from both. The TF-IDF Word2Vec augmented using Count Vector is used for the Content-Based part and the SVD-Based collaborative filtering model for the collaborative-filtering part.

The implementation is done based on the flow-diagram shown in Figure 6

7 Results

Upon running the Hybrid Algorithm, a user has to input the User ID or the Book name. Based on the input provided to the algorithm, the program will trigger content-based filtering if we have the data available for comparing the similarities for the given item name. Else, the SVD-based latent model gets triggered to handle the sparse data and make

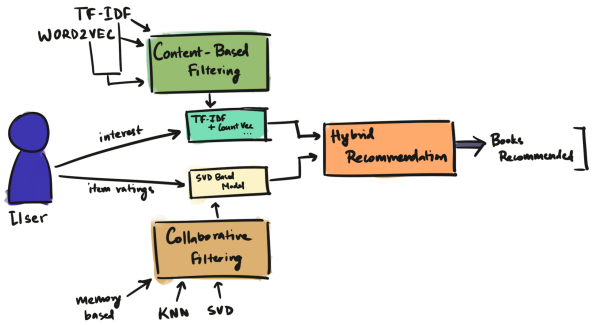
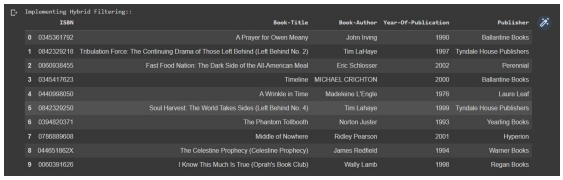


Figure 6: Hybrid Filtering Flow Diagram

recommendations accordingly. This implementation of the Hybrid model, gives us the following results-



Implementing Hybrid Filtering::

	ISBN	Book-Title	Book-Author	Year-Of-Publication	Publisher
0	0345361792	A Prayer for Owen Meany	John Irving	1990	Ballantine Books
1	0842329218	Tribulation Force: The Continuing Drama of Those Left Behind (Left Behind No. 2)	Tim LaHaye	1997	Tyndale House Publishers
2	0060938455	Fast Food Nation: The Dark Side of the All-American Meal	Eric Schlosser	2002	Perennial
3	0345417623	Timeline	MICHAEL CRICHTON	2000	Ballantine Books
4	0440998050	A Wrinkle in Time	Madeleine L'Engle	1976	Laura Leaf
5	0842329250	Soul Harvest: The World Takes Sides (Left Behind No. 4)	Tim Lahaye	1999	Tyndale House Publishers
6	0394820371	The Phantom Tollbooth	Norton Juster	1993	Yearling Books
7	0786889608	Middle of Nowhere	Ridley Pearson	2001	Hyperion
8	044651882X	The Celestine Prophecy (Celestine Prophecy)	James Redfield	1994	Warner Books
9	0060391626	I Know This Much Is True (Oprah's Book Club)	Wally Lamb	1998	Regan Books

Figure 7: Hybrid Filtering Results

With, this the hybrid model is able to provide recommendations both based on the user data and the book data, in comparison t the previous methods only working with one of these data inputs.

8 Conclusion

In this project, we successfully implemented different methods for Content-Based and Collaborative Filtering, thereby, implementing a Recommendation System as mentioned in our initial objectives in the proposal. On top of this, we extended the baseline by implementing multiple models, drawing a comparison between them and with other state-of-the-art models. Finally, we developed a hybrid book recommendation system by integrating the best result-driven approaches of content-based and collaborative filtering

In the future, we can further improve our recommendation system by taking inspiration from the latest filtering techniques such as BERT, GLOVE, Glocal K, etc) and by consolidating book datasets with meaningful features.

References

[1] C.-N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, "Improving recommendation lists through topic diversification," in *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, (New York, NY, USA), p. 22–32, Association for Computing Machinery, 2005.

[2] D. Bamman and N. A. Smith, "New alignment methods for discriminative book summarization," 2013.

[3] D. Subramanian, "Building a content-based book recommendation engine." Available at https://www.kdnuggets.com/2020/07/building-content-based-book-recommendation-engine.html.

[4] D. Subramanian, "Content-based recommendation system using word embeddings." Available at https://pub.towardsai.net/content-based-recommendation-system-using-word-embeddings-c1c15de1ef95.

[5] N. Mustafa, A. O. Ibrahim, A. Ahmed, and A. Abdullah, "Collaborative filtering: Techniques and applications," *2017 International Conference on Communication, Control, Computing and Electronics Engineering (ICCCCEE)*, pp. 1–6, 2017.

[6] J. S. Breese, D. Heckerman, and C. Kadie, "Empirical analysis of predictive algorithms for collaborative filtering," in *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, (San Francisco, CA, USA), p. 43–52, Morgan Kaufmann Publishers Inc., 1998.

[7] S. C. Han, T. Lim, S. Long, B. Burgstaller, and J. Poon, "Glocal-k: Global and local kernels for recommender systems," in *Proceedings of the 30th ACM International Conference on Information Knowledge Management*, CIKM '21, (New York, NY, USA), p. 3063–3067, Association for Computing Machinery, 2021.

[8] Y. Leng, R. Ruiz, X. Dong, and A. S. Pentland, "Interpretable recommender system with heterogeneous information: A geometric deep learning perspective," *SSRN Electronic Journal*, 2020.

[9] C. Sammut and G. I. Webb, eds., *Latent Factor Models and Matrix Factorizations*, pp. 571–571. Boston, MA: Springer US, 2010.