The Application of Deep Learning Algorithms to Computational Problems

İlker Gül ilkergul@sabanciuniv.edu
Computer Science & Engineering, 3rd Year

Metehan Koç kocmetehan@sabancıuniv.edu
Computer Science & Engineering, 2nd Year

Tahir Turgut tahirturgut@sabancıuniv.edu
Computer Science & Engineering, 2nd Year

Project Supervisor

Dr. Mehmet Keskinöz
Electronics Engineering, Sabancı University

Abstract

In order to suggest relevant products -such as books, movies, texts- lots of methods are used as recommender systems. This research focuses on which method would provide more successful book recommendations to the people. Initially, research examines cosine similarity, KNN, matrix factorization(MF) methods' background and implementation and explains other methods with those. Afterwards, these methods are compared by their error rate. Finally, successful -giving more relevant recommendations- methods of comparison are represented and further developments about that methods are suggested, and possible future work is explained.

Keywords: Deep-learning, recommender system, cosine similarity, collaborative filtering

1. Introduction

Recommender systems are programs which aim to generate meaningful recommendations to specific users for a particular product that might interest them (Melville & Sindhwani, 2010). These kinds of algorithm-based programs can be used in many industrial areas. For instance, they are being used to recommend books and CDs at Amazon.com, movies at Netflix.com and other internet sites (Liu et al., 2015).

There are 3 types of recommender systems: Content-based filtering, Collaborative (user-based) filtering, Hybrid filtering (Melville & Sindhwani, 2010). In addition to that, lots of methods/algorithms can be included into these 3 types of recommender systems. To exemplify: cosine similarity can be considered as Content-based, whereas K-nearest Neighbors (KNN) algorithm, Alternate Least Square(ALS) and matrix factorization as Collaborative. In this paper, we describe those methods and compare them in terms of success while suggesting books -by checking their error rate-. In order to increase adaptability, we use previous researches about recommendation and related algorithms, as well.

First research is about definition and implementation of cosine similarity. Gunawan, Sembiring and Budiman (2018) stresses that text relevance can be used for suggesting new web pages or documents to read to users beside the crawling method. They highlight that cosine similarity of bunch of words brings better relevance between texts. In addition to that, they provide techniques to create groups of similar texts -based on how their words are similar- by vectorizing each of them and checking their cosine similarity at the very end of the process (Gunawan et al., 2018).

Second paper gives explanations about cosine similarity and matrix factorization's positive and negative sides and implementation of the algorithm. Wen et al. (1993) claims that mixing these two algorithms into a new one called "cosine matrix factorization" (CosMF) would bring more successful results. The research explains how to implement two methods to create one hybrid method. They argue that matrix factorization method is efficient while recommending but suffers when the user's rating number is low. In order to fix it, they offer cosine similarity methods to replace that part (Wen et al., 1993).

Final academic search underlines deep learning approaches for recommender systems. According to Batmaz et al. (2018), success of deep learning and neural networks have an impact on recommender systems, as well as other industrial areas. Furthermore, they adapt deep learning techniques such as restricted boltzmann machines, deep belief networks, autoencoders and 2 types of neural networks: recurrent and convolutional. The implementation of these methods is described in depth to guide how to fit those industrial techniques to new areas of recommender systems. Finally, study shows that deep learning techniques helped to overcome 2 major challenges of recommender systems: scalability and sparsity (Batmaz et al., 2018).

The rest of the paper is organized as follows: Chapter 2 summarizes the database which is used in that research by its specialities; Chapter 3 gives an overview of usage and the results of KNN, ALS, MF and cosine similarity algorithms which are included for comparison; Chapter 4 focuses on other methods that we have worked on throughout the project period; In Chapter 5, we explained the methods which give the most precise results and should be favoured for recommending books and suggest further developments for that particular filtering concept.

2. Data Explanation

	user_id	book_id	rating	authors	original_publication_year	original_title	title	language_code	tag_name
0	1	258	5	Carlos Ruiz Zafón, Lucia Graves	2001.0	La sombra del viento	The Shadow of the Wind (The Cemetery of Forgot...	eng	to-read fantasy favorites currently-reading fi...
1	11	258	3	Carlos Ruiz Zafón, Lucia Graves	2001.0	La sombra del viento	The Shadow of the Wind (The Cemetery of Forgot...	eng	to-read fantasy favorites currently-reading fi...
2	143	258	4	Carlos Ruiz Zafón, Lucia Graves	2001.0	La sombra del viento	The Shadow of the Wind (The Cemetery of Forgot...	eng	to-read fantasy favorites currently-reading fi...
3	242	258	5	Carlos Ruiz Zafón, Lucia Graves	2001.0	La sombra del viento	The Shadow of the Wind (The Cemetery of Forgot...	eng	to-read fantasy favorites currently-reading fi...
4	325	258	4	Carlos Ruiz Zafón, Lucia Graves	2001.0	La sombra del viento	The Shadow of the Wind (The Cemetery of Forgot...	eng	to-read fantasy favorites currently-reading fi...

Table 1. 5 Rows of Goodbooks-10k with significant attributes

Goodbooks-10k dataset is used to apply and evaluate recommendation algorithms in this study. This dataset includes 10 thousand books which were published in various years between

1750 BCE and 2017 CE and about 6 million ratings of users to these books. Ratings are in scale of 1 to 5 which 1 indicates dislike, 5 indicates like of users. It also contains features of books such as author, title, publication year, language, tags which are given by users to books. Table 1 shows the first 5 rows of the dataset.

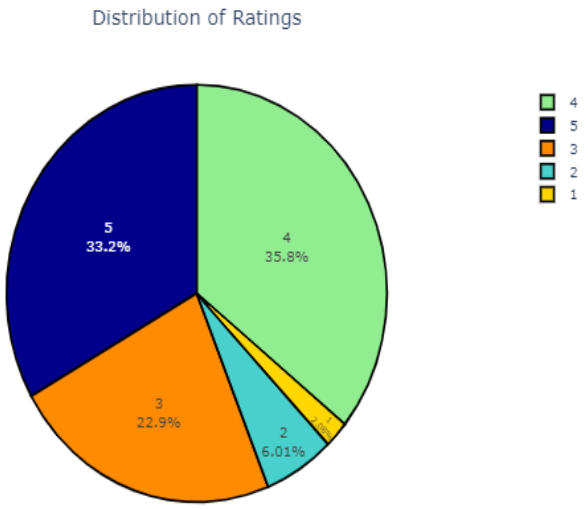

Figure1. Distribution of User Ratings in Pie Chart

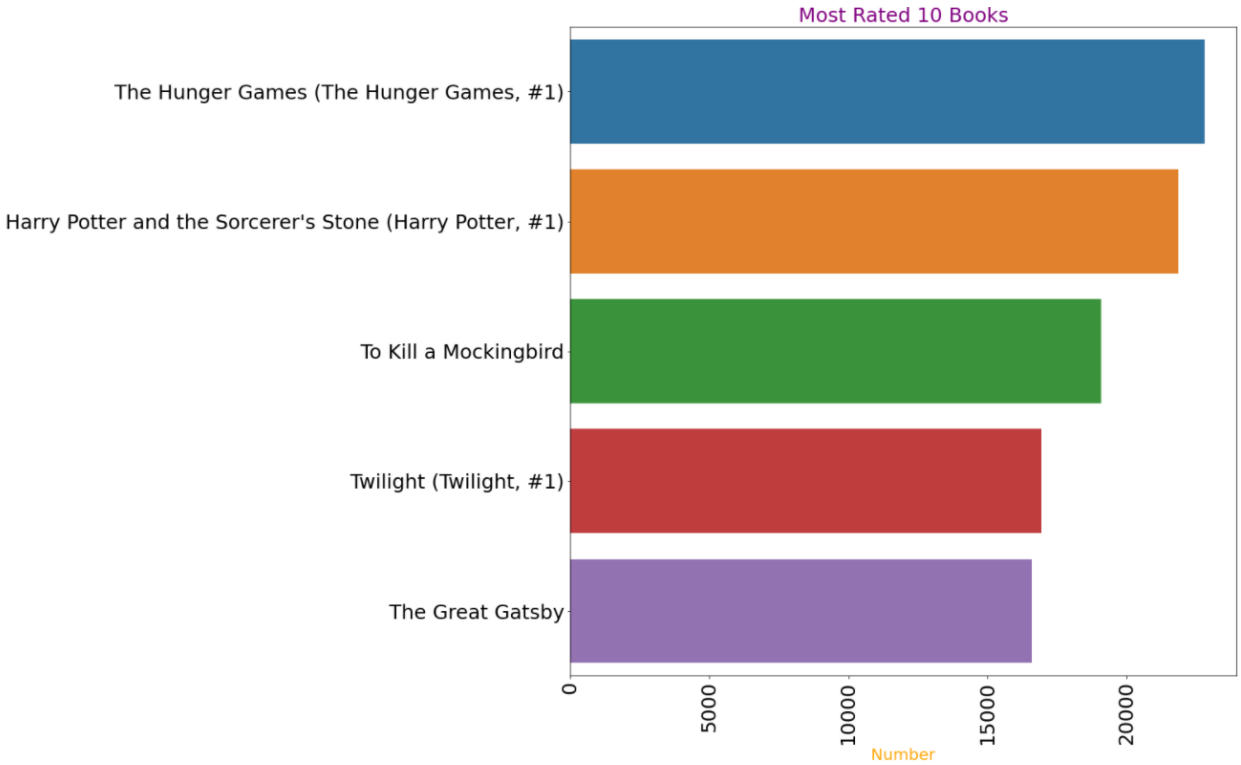

Figure 2. Most Rated 5 Books

Figure 1 shows distribution of 6 million ratings in a pie chart. Most of the ratings are distributed between 3 and 5, these ratings form approximately 92 percent of total ratings while ratings 1 and 2 only form about 8 percent of total ratings. It probably shows that books are mostly rated by users who likes these books. Figure 2 shows the most rated 5 books by users. These books are some of the most popular books around the world. It may indicate that the dataset which is used in this study mostly suits the real world, so it is a good choice to create a recommendation system with this dataset.

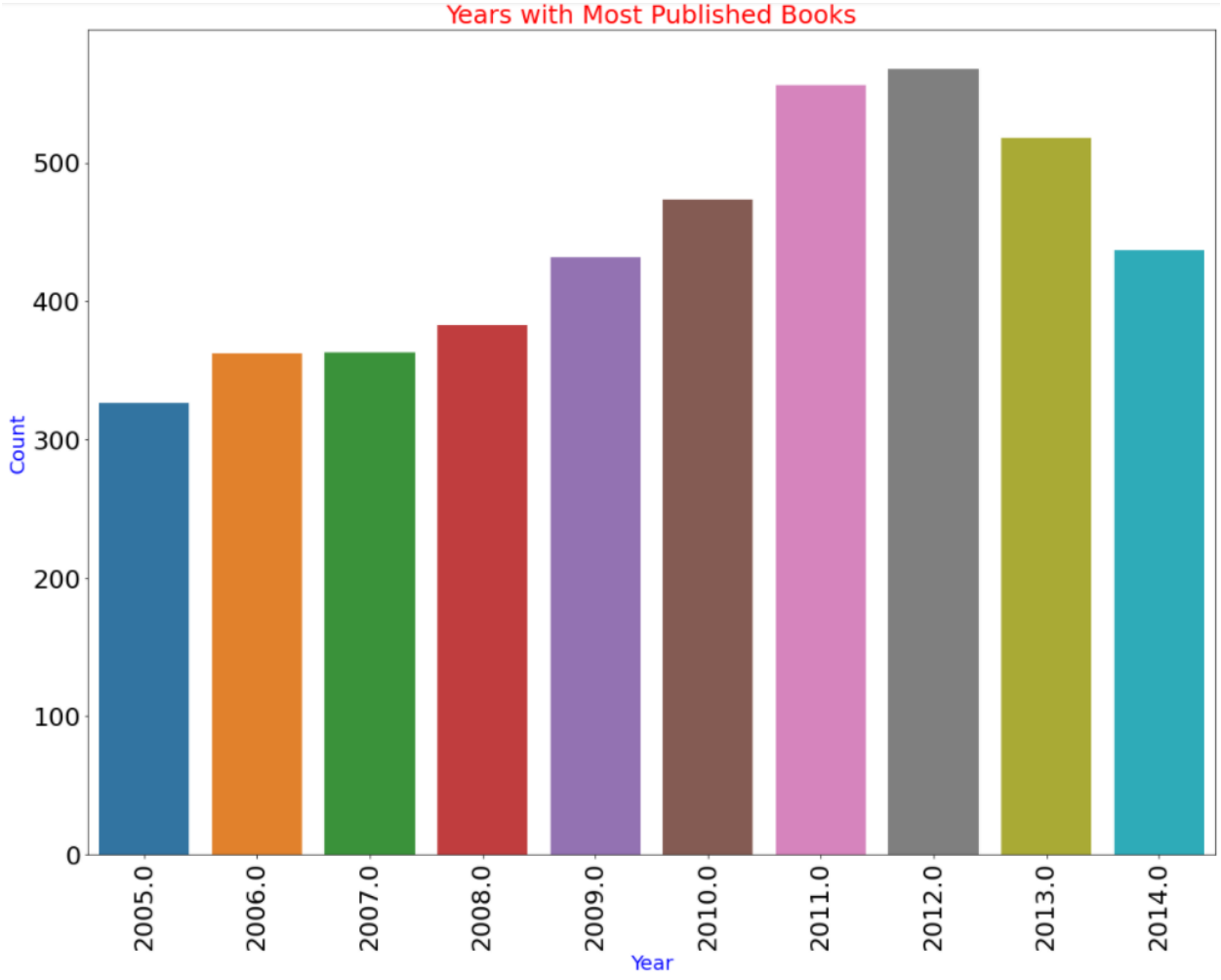

Figure 3. Years with Most Published Years

Figure 3 shows the first 10 years with most published books in goodbooks-10k dataset. This figure emphasizes that most of the books' publication years are between 2005 and 2014. It

shows that this dataset includes mostly contemporary books therefore it will be useful to create a recommendation model which can evaluate and recommend latest books to the users in the study.

Because of the computational limitations, only users which give at least 100 ratings and books which have 100 ratings as minimum are used in this study.

3. Methodology

This study is focused on content based and collaborative filtering systems. In content-based filtering, cosine similarity is used in 4 different matrices which are created by using "CountVectorizer" and "TF-IDF Vectorizer" on tag and title features of books. In collaborative filtering, three different KNN (K-Nearest Neighbors) models, SVD (Singular Value Decomposition) and K-means Clustering are used, and these models are evaluated and compared by root mean square error metric.

3.1. Content-Based Filtering

According to Robin van Meteren and Maarten van Someren (2000), content-based filtering is a system which is based on correlation between features of items and user's preferences (p. 2). There are several methods to build content-based filtering systems. In this study, cosine similarity is used on vector space of book's title and tags which is created by TF-IDF and Count vectorizer to find similarities between items and recommend them with cosine similarity accordingly.

3.1.1. Cosine Similarity by TF-IDF Vectorizer

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{10}{df_i}\right)$$

- $f_{i,j}$ = number of occurrences of i in j
- df_i = number of documents containing i
- $N = total\ number\ of\ documents$

Figure 4. General Formula of TF-IDF

TF-IDF (term frequency-inverse document frequency) is a statistical measure to show the importance of a word in a text. It is calculated by multiplication of word's frequency (TF) by inverse document frequency (IDF) which shows rareness of a word. Figure 4 shows the general formula of TF-IDF method. This method is mostly used for information retrieval and keyword extraction.

	adventure	adventurers	adventures	adversary	adversity
The Hunger Games	0.090461	0.0	0.0	0.0	0.0
Harry Potter and the Philosopher's Stone	0.045263	0.0	0.0	0.0	0.0
Twilight	0.000000	0.0	0.0	0.0	0.0
To Kill a Mockingbird	0.000000	0.0	0.0	0.0	0.0
The Great Gatsby	0.000000	0.0	0.0	0.0	0.0

Table 2. A part of Vector Space Created By TF-IDF on Tags

	adventure	adventurers	adventures	adversary	adversity
The Hunger Games	0.090461	0.0	0.0	0.0	0.0
Harry Potter and the Philosopher's Stone	0.045263	0.0	0.0	0.0	0.0
Twilight	0.000000	0.0	0.0	0.0	0.0
To Kill a Mockingbird	0.000000	0.0	0.0	0.0	0.0
The Great Gatsby	0.000000	0.0	0.0	0.0	0.0

Table 3. A part of Vector Space Created By TF-IDF on Titles

In this study, TF-IDF method is used to find the importance of books' tags and titles to compare them. For example, Table 2 shows that the "adventure" tag is one of the most significant tags for "The Hunger Games" books, when compared to "Harry Potter and the Philosopher's Stone", "adventure" tag becomes less important for this book. Similarly, Table 3

indicates that “hunger” title is one of the most important tags for both “Catching Fire” and “Mockingjay” books by using the TF-IDF method on these books’ titles.

title	The Hunger Games (The Hunger Games, #1)	Harry Potter and the Sorcerer's Stone (Harry Potter, #1)	Twilight (Twilight, #1)	To Kill a Mockingbird	The Great Gatsby	The Fault in Our Stars
Harry Potter and the Order of the Phoenix (Harry Potter, #5)	0.0	0.730191	0.0	0.0	0.0	0.0
The Lovely Bones	0.0	0.000000	0.0	0.0	0.0	0.0
Harry Potter and the Chamber of Secrets (Harry Potter, #2)	0.0	0.736109	0.0	0.0	0.0	0.0

Table 4. Cosine Similarity Matrix on Titles

title	The Hunger Games (The Hunger Games, #1)	Harry Potter and the Sorcerer's Stone (Harry Potter, #1)	Twilight (Twilight, #1)	To Kill a Mockingbird	The Great Gatsby
The Hunger Games (The Hunger Games, #1)	1.000000	0.389626	0.406857	0.233612	0.234380
Harry Potter and the Sorcerer's Stone (Harry Potter, #1)	0.389626	1.000000	0.370155	0.300862	0.286403
Twilight (Twilight, #1)	0.406857	0.370155	1.000000	0.194107	0.219647
To Kill a Mockingbird	0.233612	0.300862	0.194107	1.000000	0.835814
The Great Gatsby	0.234380	0.286403	0.219647	0.835814	1.000000

Table 5. Cosine Similarity Matrix on Tags

Cosine similarity is a metric which measures similarity between two vectors by finding the cosine of angle between these vectors. In this research, cosine similarity is used on the vector space model which is created by the TF-IDF method to find similarities between books. Table 4 shows similarities between books based on cosine similarity between titles. It emphasizes that Harry Potter books are similar because they have some common words in their titles such as “Harry”, “Potter”. As different from Table 4, Table 5 shows similarities between books by using books’ tags. According to table 5, “Twilight” is similar to “the Hunger Games”, “Harry Potter” which have higher than 0.40 cosine value and it makes sense because all of them belong to the fantasy fiction genre.

```
Recommending 5 products similar to The Hunger Games (The Hunger Games, #1)
-------
Recommended: The World of the Hunger Games (Hunger Games Trilogy) (score:0.9438314804897675)
Recommended: The Hunger Games Trilogy Boxset (The Hunger Games, #1-3) (score:0.8956104481003889)
Recommended: Catching Fire (The Hunger Games, #2) (score:0.7574537808593318)
Recommended: Mockingjay (The Hunger Games, #3) (score:0.7431803983724992)
Recommended: Hunger (score:0.7162304386557543)
```

Table 6. Recommendation on Titles

```
Recommending 5 products similar to Harry Potter and the Sorcerer's Stone (Harry Potter, #1)
-------
Recommended: Harry Potter and the Prisoner of Azkaban (Harry Potter, #3) (score:0.9840073445393256)
Recommended: Harry Potter and the Chamber of Secrets (Harry Potter, #2) (score:0.9820423548270265)
Recommended: Harry Potter and the Deathly Hallows (Harry Potter, #7) (score:0.9657235269252039)
Recommended: Harry Potter and the Half-Blood Prince (Harry Potter, #6) (score:0.9650833898658011)
Recommended: Harry Potter and the Goblet of Fire (Harry Potter, #4) (score:0.9488388043776768)
```

Table 7. Recommendation on Tags

Table 6 and table 7 shows recommendations based on cosine similarity of tags and titles. In table 6, there is only one irrelevant book called “Hunger” in recommendation because of its title. All other books are in “The Hunger Games “trilogy. In table 7, all recommendations are relevant and in the Harry Potter Series. It may show that using tags on cosine similarity is more sensible than using titles.

3.1.2. Cosine Similarity by TF-IDF Vectorizer

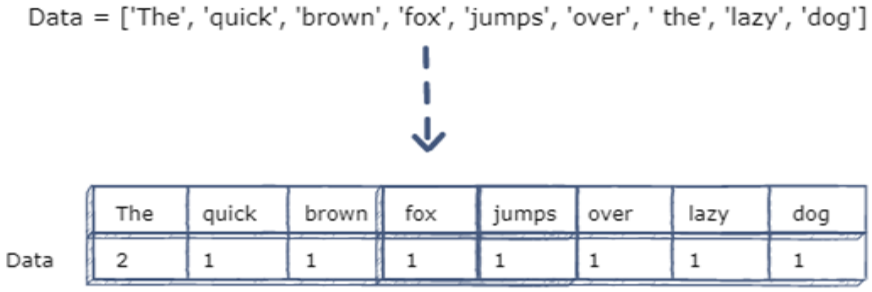

Figure 5. Application of CountVectorizer

Count vectorizer is a method which converts collection of text documents to a matrix of token count. Figure 5 shows an illustration of count vectorizer's conversion process. In this research, we applied this method to both titles and tags of books to create a vector space for cosine similarity. After that, like in TF-IDF vectorizer, cosine similarity is used on titles and tags to find similarities between books.

	book_title	sim_books	scores	words
0	Little Women (Little Women, #1)	Little Women (Little Women, #1)	1.000000	[little , women]
1	Little Women (Little Women, #1)	Little Men (Little Women, #2)	0.866025	[little , men , women]
2	Little Women (Little Women, #1)	Jo's Boys (Little Women, #3)	0.816497	[boys , little , women]
3	Little Women (Little Women, #1)	Good Wives (Little Women, #1.5)	0.816497	[good , little , women]
4	Little Women (Little Women, #1)	Crazy Little Thing (Bell Harbor, #1)	0.707107	[little]
5	Little Women (Little Women, #1)	Little Bear's Friend	0.707107	[little]
6	Little Women (Little Women, #1)	Five Little Peppers and How They Grew	0.707107	[little]
7	Little Women (Little Women, #1)	The Three Little Pigs (Disney Classic)	0.707107	[little]
8	Little Women (Little Women, #1)	The Women of Brewster Place	0.707107	[women]
9	Little Women (Little Women, #1)	Little Dorrit	0.707107	[little]

Table 8. Cosine Similarity on Titles

	book_title	sim_books	scores	tags
0	Allegiant (Divergent, #3)	Insurgent (Divergent, #2)	1.0	[adult , adult fiction , fiction , young , you...
1	Allegiant (Divergent, #3)	Harry Potter and the Chamber of Secrets (Harry...	1.0	[adult , adult fiction , fiction , young , you...
2	Allegiant (Divergent, #3)	The Lightning Thief (Percy Jackson and the Oly...	1.0	[adult , adult fiction , fiction , young , you...
3	Allegiant (Divergent, #3)	Paper Towns	1.0	[adult , adult fiction , fiction , young , you...
4	Allegiant (Divergent, #3)	City of Ashes (The Mortal Instruments, #2)	1.0	[adult , adult fiction , fiction , young , you...
5	Allegiant (Divergent, #3)	The Maze Runner (Maze Runner, #1)	1.0	[adult , adult fiction , fiction , young , you...
6	Allegiant (Divergent, #3)	The Hunger Games (The Hunger Games, #1)	1.0	[adult , adult fiction , fiction , young , you...
7	Allegiant (Divergent, #3)	Holes (Holes, #1)	1.0	[adult , adult fiction , fiction , young , you...
8	Allegiant (Divergent, #3)	Thirteen Reasons Why	1.0	[adult , adult fiction , fiction , young , you...
9	Allegiant (Divergent, #3)	The Book Thief	1.0	[adult , adult fiction , fiction , young , you...
10	Allegiant (Divergent, #3)	Harry Potter and the Prisoner of Azkaban (Harr...	1.0	[adult , adult fiction , fiction , young , you...

Table 9. Cosine Similarity on Tags

Table 8 and Table 9 shows recommendations by using cosine similarity on the vector space model which is created by the count vectorizer. Table 8's recommendations are not good recommendations for "Little Women" books because this model only considers similarities between titles. On the other hand, Table 9 shows a better recommendation than table 8 because of its usage of books' tags. , "Insurgent", "Harry Potter", "Hunger Games" are all fantasy fiction so they are similar to Allegiant.

3.2. Collaborative Filtering

Collaborative filtering methods use the ratings in the form of a matrix called the rating matrix. Rows of the rating matrix represent the ratings of different users and columns of the matrix represent the different items. The rating matrix is usually very sparse, known as sparsity problem since users rate a relatively small number of items. The main goal of the collaborative filtering approach is to predict these empty values. If empty values can be predicted successfully, items with high predicted ratings can be recommended to users.

3.2.1. KNN Based Algorithms

The K-nearest Neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems (Harrison, 2019). KNN technique uses the idea of closeness between data points. To clarify, it finds the distances between query and data points in the given data, selecting the specified number examples (K) closest to the query for labeling to classify. For KNN, we do not need to tune several parameters for the process, so it brings an advantage in usage. We used some KNN inspired algorithms to predict the rating attribute of our dataset to evaluate our algorithms in which we can improve algorithms that are applicable for a recommendation.

By using different KNN inspired algorithms, we want to find which one is more eligible for our dataset. KNNBaseline is a KNN filtering algorithm that focuses on baseline ratings. KNNBasic as it is clear with its name uses the basic elements of the KNN technique. Furthermore, KNNWithMeans uses the ratings of each user in our dataset to apply a filtering process. We have used the same data preprocessing for each algorithm. So, there should not be a

positive tendency for any algorithm. By applying Root Mean Square Error(RMSE), which is a frequently used measure technique to see the residual deviation of a dataset, we have come up with some results to understand which inspired algorithm fits best for our research data. The results will be shown in the below table.

Algorithms	RMSE Results
KNNBaseline	0.805
KNNBasic	0.832
KNNWithMeans	0.807

Table 10: KNN Inspired Algorithms Results

As it is seen in the results, RMSE scores are very close. We can say for our dataset there is no big difference occurred between algorithms. However, KNNBaseline has a better deviation residual than other algorithms. Hence, it is advantageous to proceed with KNNBaseline in a further process to improve our prediction while using a KNN technique.

Furthermore, we also applied a book recommendation with our KNN model. In the figure below, you will see a list of book recommendations for the selected book, which is “The Museum of Innocence”.

```
[ ] def recommendBook(book_id):
      raw_id = book_id
      inner_id = trainset.to_inner_iid(raw_id)
      neighbors = algo.get_neighbors(inner_id, k=10)
      neighbors = (trainset.to_raw_iid(inner_id)
                   for inner_id in neighbors)
      print(('Top recommendations for {} are:').format(books.set_index("book_id").loc[raw_id].title))
      for i in neighbors:
        print(books.set_index("book_id").loc[i].title)

[ ] recommendBook(book_id=5851)

    Top recommendations for The Museum of Innocence are:
    Snow
    Never Let Me Go
    Norwegian Wood
    Gone with the Wind
    Mockingjay (The Hunger Games, #3)
    My Name is Red
    Alice's Adventures in Wonderland & Through the Looking-Glass
    Harry Potter and the Sorcerer's Stone (Harry Potter, #1)
    The Ocean at the End of the Lane
    Rebecca
```

Figure 6: Sample of the KNN Code for Recommended Books

We examined the list of books recommended by our model. Some recommended books are from the same author and genre. However, some books, such as but not limited to "Mockingjay(The Hunger Games, #3) and "Harry Potter and the Sorcerer's Stone(Harry Potter, #1), in the recommendation list is not related to the genre of the selected book for the recommendation. We mainly think it is due to popularity bias which is a popular limitation while using KNN models. It occurs due to the lack of customization of interactions since the model looks for the frequency of the interactions. Another aspect that causes these unfamiliar book recommendations can be the "item cold-start problem". This limitation is also as frequent as popularity bias in KNN-based collaborative filtering recommender models (Subramaniyaswamy & Logesh, 2017).

3.2.2. Matrix Factorization

Matrix factorization (MF) is another collaborative filtering method which has been successful in many applications such as Netflix Prize. It is found to be effective in reducing the impacts of sparsity problem. MF identifies the hidden features of users and items as vectors using the rating matrix. It decomposes the rating matrix into two matrices such that the inner product of these matrices estimates the rating matrix. These two matrices capture the latent features of users and items, respectively.

Suppose there are N users and M items. Then the rating matrix has the shape *NxM* and matrix decomposition produces the user latent matrix with shape *NxD* and item latent matrix with shape *MxD* where D is the number of latent factors. D is a hyperparameter that is smaller than N and M. Increasing D improves the accuracy of the model but increases the time complexity. Model can also overfit if D is too high. There are many decomposition methods for matrix factorization, most famous ones are singular value decomposition(SVD), non-negative matrix factorization(NMF) and probabilistic matrix factorization(PMF).

SVD method initializes the latent matrices randomly, then it predicts the ratings by taking the dot product of these matrices and adding the user and item biases. SVD decomposes the matrix by minimizing the squared error of the predictions. It can also include regularization terms. The minimization is usually performed by simple stochastic gradient descent(SGD). RMSE score of SVD method is 0.841 for our dataset.

PMF method is very similar to SVD. This method doesn't include user and item biases in the prediction process. RMSE score of this method is 0.835 for our dataset.

NMF method sets the predictions as the dot product of the latent matrices where user and item factors are kept positive. Optimization is also done using SGD with an additional step to ensure the non-negativity of factors. RMSE score of NMF is 0.866 for our dataset.

All the matrix factorization algorithms have been implemented using the Surprise package. PMF is implemented as an unbiased version of SVD. The number of latent factors are set to 100 for SVD and PMF and 15 to NMF. Algorithms are evaluated using the cross-validation method of the same package.

3.2.3. Alternate Least Square

In principle, the algorithm of the alternating least squares (ALS) factorizes a given matrix A into two B and C variables, so that A ≈ BTC is concerned. The unknown dimensions of the row are given to the algorithm as parameters and are called latent factors (Liao, 2018).

In our dataset, we used the ALS method for the "ratings". It gave us flexibility to manipulate the algorithm and provided a solving scalability and sparseness for the selected attribute of the dataset. We used it for prediction to test the accuracy of our model. To reach a better result, we tuned its hyperparameters via hold-out validation and cross validation. After evaluations, we reached a RMSE score of 0.90. In the following table there is a demonstration of our result

User_id	Book_id	Rating	Prediction	Title
23007	6654	5	4.789	The Endurance: Shackleton's Legendary Antarctic Expedition
25645	6654	5	4.158	The Endurance: Shackleton's Legendary Antarctic Expedition
40904	6654	4	3.790	The Endurance: Shackleton's Legendary Antarctic Expedition
46440	6654	5	2.888	The Endurance: Shackleton's Legendary Antarctic Expedition
12056	6654	3	4.142	The Endurance: Shackleton's Legendary Antarctic Expedition

Table 11. ALS results

Since the ratings of our dataset is scaled between 1 to 5, having a RMSE of 0.90 is not a proper result to implement. Hence, we are trying to improve our result by looking at the aspects we can manipulate to get a proper result.

4. Related Work

4.1. Clustering

Clustering is the method of grouping data points into a number of categories, in such a manner that data points in the same categories are more similar than those to other data points of the same category. This is a standard technique that is done during an exploratory analysis of the results. There are many algorithms so that you can do the clustering process effectively. We have used the K-means clustering algorithm for our implementation process.

In the preprocessing period to have eligible data points, we initially create a sparse matrix. In this sparse matrix, we also add some filtering process to have a better solution with avoiding the outliers in the dataset. We did not use books that have fewer than 250 ratings and

drop users that have given fewer than 80 ratings of these most-rated books. Then, we get a sparse matrix that includes the filtered dataset with 39916 rows and 4956 columns.

In fact, to eliminate the possibility of memory issues being lost due to random access memory(RAM), we used the technique of truncated SVD to decrease the 4956 features to 200. After that, we applied the K-means clustering algorithm to cluster our dataset into 5 clusters. We decided to use 5 by checking the silhouette score of each number between 3 and 15. To explain, the silhouette score is a metric often used to determine the correct size for clustering. The best result we get by checking the silhouette score is given by K-Means clustering is with 5 clusters. It is quite arduous to visualize the clusters while having 200 composite features. However, we picked the 2 most important composite elements to get a presentation and get the perspective right.

Figure 7: K-means clustering

In the figure above, we can observe that the clusters we have are not well separated for 2 clusters represented by yellow and purple. It is mainly due to the selection of only 2 parameters

for the visualization. However, since these two features are the most significant features in the clustering process, we can improve our clustering algorithm to have a better visualization of the clusters. We also observe that by filtering the matrix we have used we got a few data points that can be labeled as outliers. In addition to that, we can manipulate our algorithm by changing the number of total composite features and tuning some parameters in the K-means clustering algorithm to have a better result.

4.2. Seq2seq

Seq2seq is a machine learning algorithm that is developed by Google that is commonly used for language processing purposes. It is an algorithm that uses a deep learning model which is a recurrent neural network (RNN). To decrease the risk of vanishing gradient which is a machine learning error that is faced in training processes while using artificial neural network models, long short-term memory(LSTM) is frequently used.

For our dataset, first, we implement a tag matrix by using the “tags” attribute. Then, we filter tags that would give improper results in further processes. After that, we scaled the tag matrix by using the TF-IDF vectorizer which we have used in the cosine similarity approach.

Then, we have tried to add our seq2seq model to our recommendation algorithm after removing the accountability of popularity to get an unbiased result. We are currently working on getting a recommendation result with the seq2seq model by improving the model that we have designed.

5. Conclusion and Future Work

5.1. Visional Improvement

The importance of recommender systems and their real-life implementations were studied during the project. Several machine learning algorithms such as KNN and K-means have been studied, engineered, and applied to our dataset. Huge datasets were treated with preprocessing and cleaning of data. Collaborative filtering and content-based filtering have been researched intensively via reading journal and several academic papers.

Themes relating to deep learning implementation and how it can be achieved were explored and analyzed during the study. Teamwork and weekly role delivery were done, and academic speech skills were improved. This also acknowledged the potential of machine learning and deep learning systems and the need to improve language recognition.

5.2. Conclusion and Future Work

Recommender systems help people to find the most suitable products when they require guidance. Throughout the process, we applied several different models to manipulate the data to give a better understanding of how models can affect the accuracy of the findings. After a lot of time spent on research, it is frankly stated there are a lot of aspects that can be examined and improved to have better results. By adapting content-based filtering and collaborative filtering techniques to the data in which the report is concentrated, we have investigated the similarity and RMSE scores to find the most elaborate technique we have implemented in this period.

Before getting in detail of the comparison of applied algorithms, deep learning and language processing are two significant figures that can be searched on and can be implemented to have better recommendation lists to bookworms. Recurrent Neural Networks based on “tags” attributes should be taken into consideration for getting more complex analysis. By developing the seq2seq implementation, better results can be provided to the users. Several language processing methods, such as extracting linguistic elements, or word embedding, can also be implemented. Besides, the classification of naive Bayesians can be researched and adapted to the dataset, to improve our understanding of the classification of the data points by having different implemented techniques.

Content-based filtering and collaborative filtering have used different attributes of the data. For the “tags” attribute and “title” attribute, language processing was needed to evaluate. By using tf-idf vectorizer and CountVectorizer we have used cosine similarity in two separate applications. Both approaches came up with a proper recommendation list. However, the tf-idf vectorizer came up with a more accurate classification since, after 5 books in the recommendation lists, CountVectorizer offered books that are not as much related as the tf-idf vectorizer. Both approaches can be improved by some tuning and filtering to have better recommendations.

Furthermore, we have applied KNN based algorithms, Matrix factorization, and Alternate Least Square Method for comparison. The least favorable approach is Alternate Least Square Method since its Root Mean Square Error(RMSE) is higher than other approaches. Since we need less deviation to have a better performance it did not give a good performance as much as others. However, it can be adapted more to our dataset and change some aspects of the algorithm if needed to develop its RMSE result.

Moreover, KNN based algorithms, as we have stated before in which RMSE scores are very close, KNNBaseline gave a higher performance than others. In matrix factorization, we have used several techniques to see the changes in RMSE. RMSE scores are between 0.83 and 0.86. It should be noted that ratings are between 1 and 5. Hence, it is needed to be worked on to decrease RMSE scores between 0.3 and 0.5 to get more accurate recommendations based on ratings.

In conclusion, the deep learning field is an ocean waiting to be discovered. Recommender systems are in trend and there are a lot of aspects that need to be improved. Deep learning techniques and language processing will help recommender systems to work on more complex datasets, better capture the user's needs, and increase the satisfaction of users. Hybrid filtering composed of content-based and collaborative filtering will enable developers to reach people more in real-life applications.

References

Batmaz, Z., Yurekli, A., Bilge, A., & Kaleli, C. (2019). A review on deep learning for recommender systems: challenges and remedies. *Artificial Intelligence Review*, *52*(1), 1-37.

Gunawan, D., Sembiring, C. A., & Budiman, M. A. (2018, March). The implementation of cosine similarity to calculate text relevance between two documents. In *Journal of Physics: Conference Series* (Vol. 978, No. 1, p. 012120).

Harrison, O. (2019, July 14). *Machine Learning Basics with the K-Nearest Neighbors Algorithm*. Medium. https://towardsdatascience.com/machine-learning-basics-with-the-k-nearest-neighbors-algorithm-6a6e71d01761

Liao, K. (2018, November 19). *Prototyping a Recommender System Step by Step Part 2: Alternating Least Square (ALS) Matrix Factorization in Collaborative Filtering*. Medium. https://towardsdatascience.com/prototyping-a-recommender-system-step-by-step-part-2-alternating-least-square-als-matrix-4a76c58714a1

Liu, R. R., Jia, C. X., Zhou, T., Sun, D., & Wang, B. H. (2009). Personal recommendation via modified collaborative filtering. *Physica A: Statistical Mechanics and its Applications*, *388*(4), 462-468.

Melville, P., & Sindhwani, V. (2010). Recommender systems. *Encyclopedia of machine learning*, *1*, 829-838.

Subramaniyaswamy, V., & Logesh, R. (2017). Adaptive KNN based recommender system through mining of user preferences. *Wireless Personal Communications*, *97*(2), 2229-2247.

Y. Koren, R. Bell, and C. Volinsky, ‘‘Matrix factorization techniques for recommender systems,’’ Computer, vol. 42, no. 8, pp. 30–37, Aug. 2009.

Wen, H., Ding, G., Liu, C., & Wang, J. (2014, September). Matrix factorization meets cosine similarity: addressing sparsity problem in collaborative filtering recommender system. In *Asia-pacific web conference* (pp. 306-317). Springer, Cham.