Movie Recommendation System Using Neural Network Based Hybrid Ensemble of Various Filtering Techniques and Score Prediction

Report submitted in fulfillment of the requirements
for the Data Mining Project

by

Shayak Das
18075056

Romit Mondal
18075051

Archit Saha
18075010

Radhika Singh
18075046

Under the guidance of

Prof. Bhaskar Biswas

Department of Computer Science and Engineering

INDIAN INSTITUTE OF TECHNOLOGY (BHU) VARANASI
Varanasi 221005, India

Dedicated to

our parents, teachers and professors for their constant guidance and support in accomplishing the task.

Declaration

I certify that

1. The work contained in this report is original and has been done by myself and the general supervision of my supervisor.

2. The work has not been submitted for any project.

3. Whenever I have used materials (data, theoretical analysis, results) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references.

4. Whenever I have quoted written materials from other sources, I have put them under quotation marks and given due credit to the sources by citing them and giving required details in the references.

Place: IIT (BHU) Varanasi

Date:21/Nov/2020

Romit Mondal
Shayak Das
Archit Saha
Radhika Singh

B.Tech. Student
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Certificate

This is to certify that the work contained in this report entitled ***'Movie Recommendation System Using Neural Network Based Hybrid Ensemble of Various Filtering Techniques and Score Prediction'*** *being submitted by* ***Archit Saha (Roll No. 18075010)*** *,****Romit Mondal (Roll No. 18075051)****,* ***Shayak Das (Roll No. 18075056)****,* ***Radhika Singh (Roll No. 18075046)****, carried out in the Department of Computer Science and Engineering, Indian Institute of Technology (BHU) Varanasi, is a bona fide work of our supervision.*

Place: IIT (BHU) Varanasi
Date:21/Nov/2020

Prof. Bhaskar Biswas
Department of Computer Science and Engineering,
Indian Institute of Technology (BHU) Varanasi,
Varanasi, INDIA 221005.

Acknowledgments

We would like to extend our gratitude to Prof. Bhaskar Biswas to give us the opportunity to do this Project .

Place: IIT (BHU) Varanasi

Date: 21/Nov/2020

Shayak Das

Romit Mondal

Archit Saha

Radhika Singh

Abstract

Recommendation systems help users by providing useful suggestions, thus reducing their search time. These recommendations can be generated in various ways like content-based, collaborative filtering, hybrid methods and other approaches. In this project, a movie dataset has been used which contains data for a number of movies. In this project, we are going to use data mining techniques to find useful correlations between the movies and recommend movies based on a given movie. We will use a movie database from TMDB 5000 Movies Dataset from Kaggle. The analysis of attributes of these movies will help us identify the decisive factors and identify user preferences accurately.

Contents

List of Figures

Chapter 1

Introduction

1.1 Overview

We will use TMDB 5000 Movies Dataset from Kaggle throughout our Project.

The rapid growth of data collection has led to a new era of information. Data is being used to create more efficient systems and this is where Recommendation Systems come into play. Recommendation Systems are a type of information filtering systems as they improve the quality of search results and provides items that are more relevant to the search item or are related to the search history of the user.

They are used to predict the rating or preference that a user would give to an item. Almost every major tech company has applied them in some form or the other: Amazon uses it to suggest products to customers, YouTube uses it to decide which video to play next on autoplay, and Facebook uses it to recommend pages to like and people to follow. Moreover, companies like Netflix and Spotify depend highly on the effectiveness of their recommendation engines for their business and success.

Data is being used to create more efficient systems, and this is where Recommendation Systems come into play. Recommendation Systems are a type of information filtering systems as they increase the quality of search results and provide details that are more relevant to the search item.

1.2 Motivation of the Research Work

It becomes necessarily important for any director to predict the success rate of his movie before it is launched to see how it will fare in real life. This Data Mining Model takes into consideration the above fact and implements a Success Rate Prediction Model based on other movies of similar types. Also , a Movie Recommender Model based on various important features is implemented to find similar movies.

Chapter 2

Methods and models

2.1 Data Pre Preprocessing

Some of the Important Attributes in the Dataset are :

- Overview: Description of the movie can suggest the type of movie and give us the keywords in a movie for example say we have Superman, the movie is based on a comic so most likely comic will be a keyword in the overview.
- Genres: Genre is very indicative of the taste of the user and thus must be used.
- Keywords: These are keywords pre existing in the data that can be said to be tags representing a movie.
- Actors: A person watches a movie if he likes a particular actor and that is why we need to include main actors but not all actors as supporting actors are not much of interest.
- Director: People often go for watching a movie based on the director as a director who has a good reputation in the industry is more sort after.
- VoteAverage: The score of the movie and how much it has been rated overall.

- Budget: The budget involved in the production of the movies.
- Revenue: The total revenue generated from the movie as a return.
- Popularity: These are the scores based on how popular a movie has been.

Keywords will play an important role in the functioning of the engine. Indeed, recommendations will be based on similarity between films and to gauge such similarities, I will look for films described by the same keywords. Hence, the content of the plot keywords variable deserves some attention since it will be extensively used.

Grouping by roots : I collect the keywords that appear in the plot keywords variable. This list is then cleaned using the NLTK package. Finally, I look for the number of occurrence of the various keywords.
We are replacing words by their main forms , keywords by their most frequent occurrence .
Groups of synonyms : I clean the list of keywords in two steps. As a first step, I suppress the keywords that appear less than 5 times and replace them by a synonym of higher frequency. As a second step, I suppress all the keywords that appear in less than 3 films .

Handling Missing Keywords : I try to fill missing values in the plot keywords variable using the words of the title. To do so, I create the list of synonyms of all the words contained in the title and I check if any of these synonyms are already in the keyword list. When it is the case, I add this keyword to the entry.
Handling Other Missing Values : I had a look at the correlation between variables and found that a few of them showed some degree of correlation, with a Pearson's coefficient ¿ 0.5. I will use this finding to fill the missing values of the gross, num

```
"outlander                    " in keywords list -> False 0
"extraterrestrial             " in keywords list -> True 4
"extraterrestrial being       " in keywords list -> False 0
"foreigner                    " in keywords list -> False 0
"stranger                     " in keywords list -> True 7
"noncitizen                   " in keywords list -> False 0
"alien                        " in keywords list -> True 80
"unknown                      " in keywords list -> False 0
```

```
narcism      -> narcissism   (init: [('narcissism', 1), ('narcism', 1)])
apparition   -> shadow       (init: [('shadow', 3), ('phantom', 3), ('apparition', 1)])
macao        -> macau        (init: [('macau', 1), ('macao', 1)])
regent       -> trustee      (init: [('trustee', 1), ('regent', 1)])
civilization -> culture      (init: [('culture', 2), ('civilization', 1)])
ark          -> ark of the covenant (init: [('ark of the covenant', 2), ('ark', 1)])
automaton    -> zombie       (init: [('zombie', 45), ('robot', 27), ('automaton', 1)])
------------------------------------------------------------------------------------
---
The replacement concerns 5.96% of the keywords.
```

Figure 2.1 Groups of synonyms

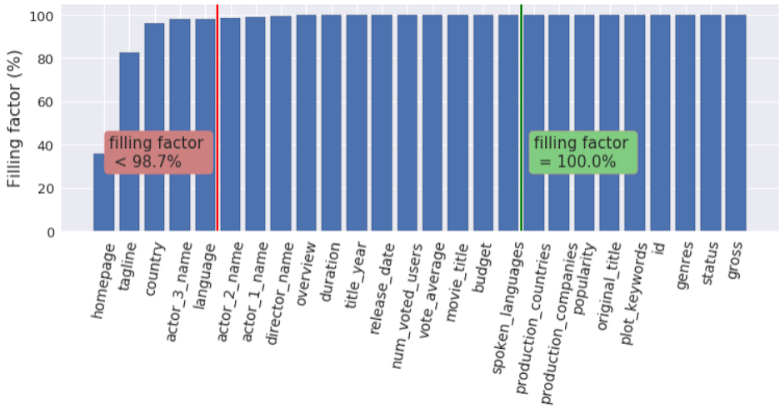


Figure 2.2 plot of Filling Factor vs Attributes

critic for reviews, num voted users , and num user for reviews variables. To do so, I will make regressions on pairs of correlated variables. First, I define a function that impute the missing value from a linear fit of the data. This function takes the dataframe as input, as well as the names of two columns. A linear fit is performed between those two columns which is used to fill the holes in the first column that was given.

Fig 2.2 shows the plot of Filling Factor vs Attributes

JSON to String : The next job is to convert the data in the before mentioned columns from json to string format. This step of preprocessing is very essential in order to extract the useful information from the data.

One Hot Encoding : One hot encoding is a process by which categorical variables are converted into a form that could be provided to data processing paradigms to do a better job. For example, the name of directors in string format can't be given as an input since this doesn't make much sense. But we can convert each director's name to an integer value and convert that to a binary array which tells which one is the director.

Data Transformation: This step is taken in order to transform the data in appropriate forms suitable for Mining process. This involves the following ways:

Normalisation: It is done in order to scale the data values in a specific range (-1.0 to 1.0 or 0.0 to 1.0)

Attribute Selection: In this strategy, new attributes are constructed from the given set of attributes to aid the mining process.

Data Reduction: The most common Data Reduction way is : Dimensionality Reduction: This reduces the size of data by encoding mechanisms. It can either be lossy or lossless. If after reconstruction from compressed data, original data can be retrieved, such reduction is named lossless reduction else it is called lossy reduction. The two effective methods of dimensionality reduction are: Wavelet transforms and PCA (Principal Component Analysis).

2.2 Data Visualitation

To achieve Data Pre-Processing, we used use Data Visualization and Exploratory Data Analysis on the given Dataset. That includes eliminating highly linearly dependent attributes or redundant attributes by checking the correlation matrix.

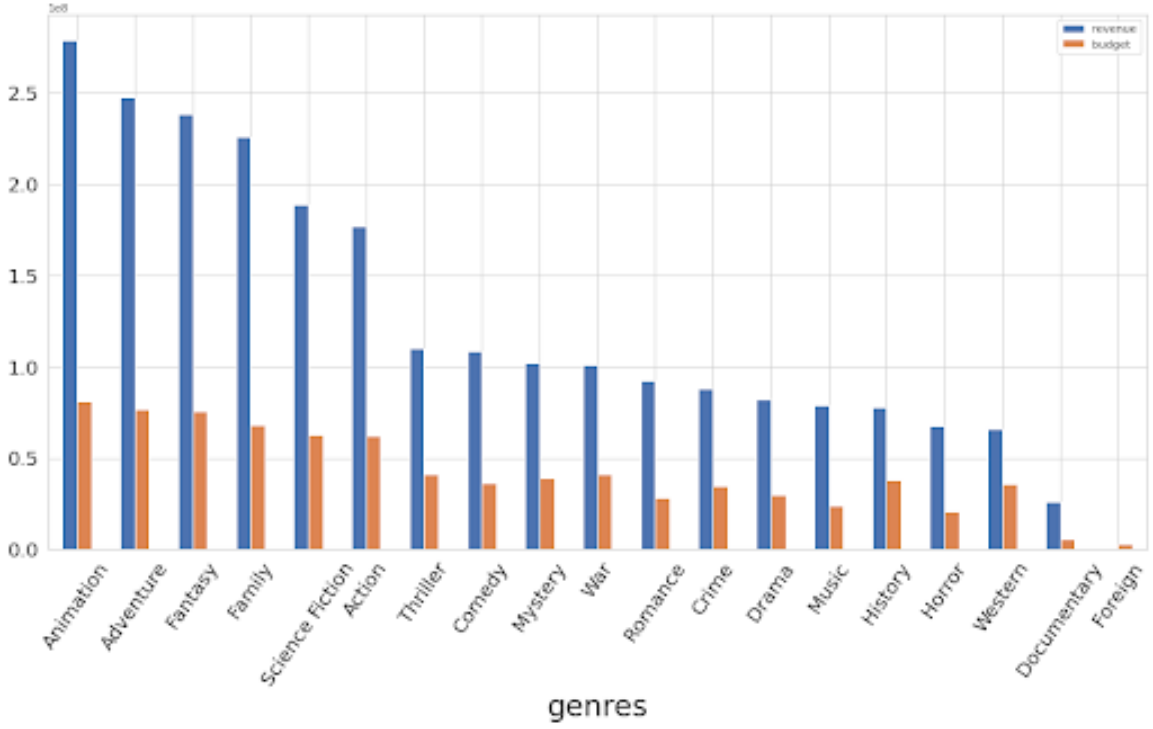
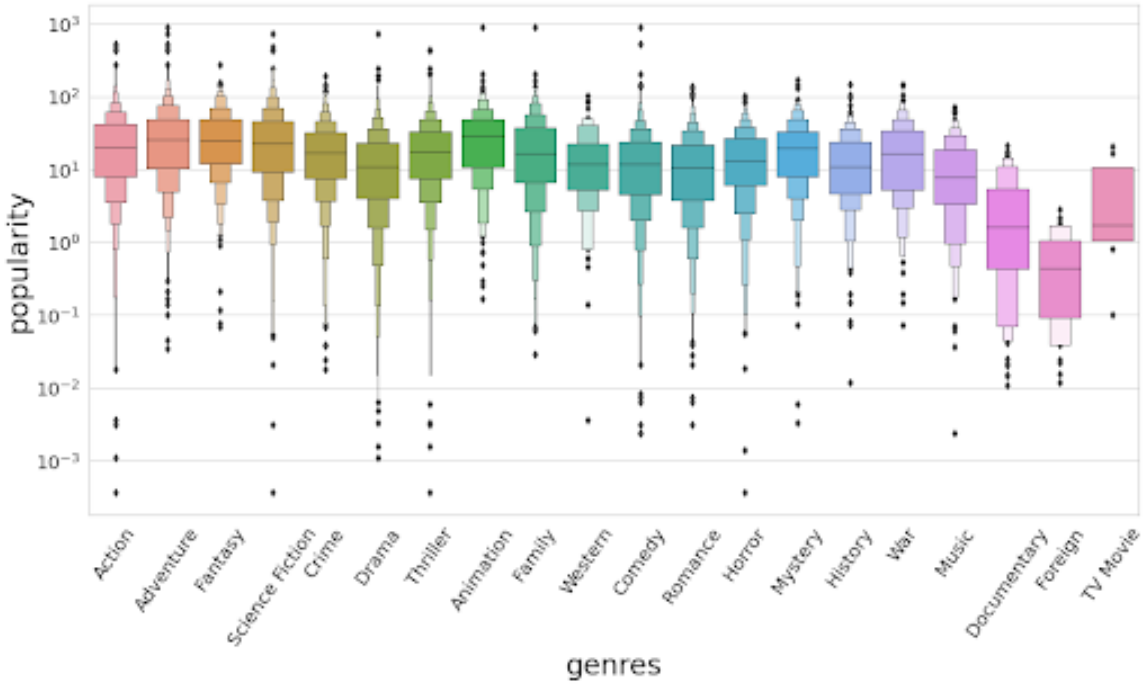
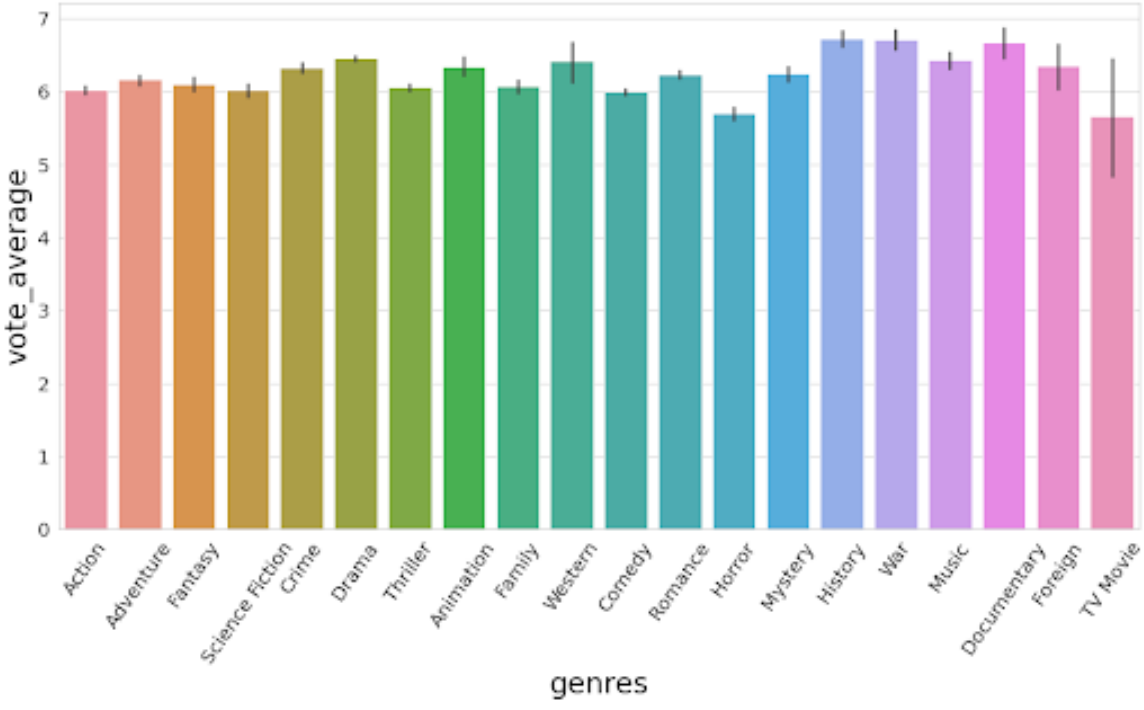


Figure 2.3 Revenue vs Budget



[fig 1]



[fig 2]

Figure 2.4 Various Data Visualization Plots

2.3 Score Prediction

We build a 1-D Neural Network which takes the data of genres, cast, director, keywords, popularity, budget and revenue as input and is trained to predict the vote average score. The feed-forward Neural Network is built using a hybrid of CNN and MLP using Convolution, ReLU Activation, MaxPool and Dense layers. We used mean squared logarithmic error as the loss function and cosine proximity as the evaluation metric.

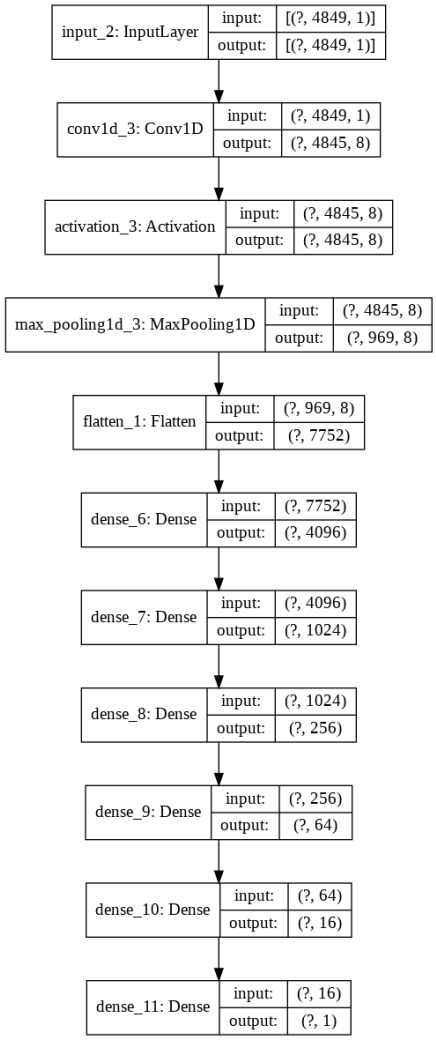


Figure 2.5 Hybrid Neural Network Model

2.4 Movie Recommendation

It is often referred to as recommender systems, a simple algorithm that aims to provide relevant and accurate information to users by filtering useful information from large data sets. The recommender system discovers information patterns in the data set by learning about consumer choices and generating results relevant to their needs and interests.

We have used collaborative content filtering to create a list of recommendations:

Content-based Filtering Systems:

In content-based filtering, items are recommended based on comparisons between item profile and user profile. A user profile is content that is found to be relevant to the user in the form of keywords(or features). A user profile might be seen as a set of assigned keywords (terms, features) collected by algorithm from items found relevant (or interesting) by the user. A set of keywords (or features) of an item is the Item profile. For example, consider a scenario in which a person goes to buy his favorite cake 'X' to a pastry. Unfortunately, cake 'X' has been sold out and as a result of this the shopkeeper recommends the person to buy cake 'Y' which is made up of ingredients similar to cake 'X'. This is an instance of content-based filtering.

We have be used the cosine similarity to calculate a numeric quantity that denotes the similarity between two movies. We use the cosine similarity score since it is independent of magnitude and is relatively easy and fast to calculate. Mathematically, it is defined as in Fig 3.7

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Figure 2.6 cosine similarity score formula

While building the recommendation engine, we are quickly faced to a big issue: the existence of sequels make that some recommendations may seem quite dumb ... As an example, somebody who enjoyed "Pirates of the Caribbean: Dead Man's Chest" would probably not like to be advised to watch its previous sequels. Hence, I tried to find a way to prevent that kind of behaviour and I concluded that the quickest way to do it would be to work on the film's titles. To do so, I used the fuzzywuzzy package to build the remove_sequels() function. This function defines the degree of similarity of two film titles and if too close, the most recent film is removed from the list of recommendations. Below is the output when del_sequels is turned off and on .

Figure 2.7 output when del-sequels is turned off and on

KNN for Movie Recommendation:

We use K Nearest Neighbours for recommending movies based on similarity scores, based on the Cosine Similarity Scores, of the selected movie with all other movies in the dataset[1]. The K closest movies based on the selected features are shown as recommended movies.

In the figure 3.9, K is 10 and the movie selected is The Godfather: Part III.

We predict the scores of the given movie along with the movie recommendations, which turns out to be the measure to see how accurate the model works.

Actual Rating: This is the actual rating which is the data given in "vote-average" column. This is the base rating, which is used as a measure to understand the accuracy of movie recommendation.

Average Rating: This is the predicted rating which we get by simply calculating the average of the rating of the recommended movies, which turns out to be quite accurate due to cosine similarity of movies.

Cumulative Mean Rating: This is the predicted rating which we get by calculating the cumulative mean of the rating of the recommended movies with the inverse of distance calculated using cosine similarity scores.

Artificially Learned Rating: This is the predicted rating which we get from the Hybrid Neural Network rating predictor, which also succeeds in predicting the score quite accurately.

```
predict_score('Godfather', 10)
```

```
Selected Movie:  The Godfather: Part III

Recommended Movies:

The Rainmaker | Genres: 'Crime','Drama','Thriller' | Rating: 6.7
The Godfather | Genres: 'Crime','Drama' | Rating: 8.4
The Cotton Club | Genres: 'Crime','Drama','Music','Romance' | Rating: 6.6
Apocalypse Now | Genres: 'Drama','War' | Rating: 8.0
New York Stories | Genres: 'Comedy','Drama','Romance' | Rating: 6.2
Peggy Sue Got Married | Genres: 'Comedy','Drama','Fantasy','Romance' | Rating: 5.9
End of Watch | Genres: 'Crime','Drama','Thriller' | Rating: 7.2
Only God Forgives | Genres: 'Crime','Drama','Thriller' | Rating: 5.6
Hannibal Rising | Genres: 'Crime','Drama','Thriller' | Rating: 6.0
Savages | Genres: 'Crime','Drama','Thriller' | Rating: 6.2


The predicted rating for The Godfather: Part III is: 6.680000
The cumulative mean rating for The Godfather: Part III is: 6.866452
The learned rating for The Godfather: Part III is: 6.226976
The actual rating for The Godfather: Part III is 7.100000
```

Figure 2.8 Movie Recommendation using KNN

Content-Based Recommender (Text Mining) We will compute pairwise similarity scores for all movies based on their plot descriptions and recommend movies based on that similarity score. The plot description is given in the overview feature of our dataset. Inverse Document Frequency is the relative count of documents containing the term is given as log(number of documents/documents with term) The overall importance of each word to the documents in which they appear is equal to TF * IDF . This will give you a matrix where each column represents a word in the overview vocabulary (all the words that appear in at least one document) and each row represents a movie, as before.This is done to reduce the importance of words that occur frequently in plot overviews and therefore, their significance in computing the final similarity score. The motivation behind IDF(inverse document frequency) is that all the words in a document are not necessarily useful in modelling the topics for that document.

Formula used for calculating the IDF is :- np.log10(number of documents/number of

documents containing that word) Since we have used the TF-IDF vectorizer, calculating the dot product will directly give us the cosine similarity score. Therefore, we will use sklearn's linear-kernel() instead of cosine-similarities() since it is faster.

```
get_recommendations('The Dark Knight Rises')
```

```
65                           The Dark Knight
299                           Batman Forever
428                           Batman Returns
1359                                 Batman
3854    Batman: The Dark Knight Returns, Part 2
119                           Batman Begins
2507                              Slow Burn
9           Batman v Superman: Dawn of Justice
1181                                  JFK
210                           Batman & Robin
Name: title, dtype: object
```

Figure 2.9 Text mining output

On calling the function with a particular movie name , it gives the following output:(Fig: 3.10)

While our system has done a decent job of finding movies with similar plot descriptions, the quality of recommendations is not that great. "The Dark Knight Rises" returns all Batman movies while it is more likely that the people who liked that movie are more inclined to enjoy other Christopher Nolan movies. This is something that cannot be captured by the present system.

Collaborative Filtering : Our content based engine suffers from some severe limitations. It is only capable of suggesting movies which are close to a certain movie. That is, it is not capable of capturing tastes and providing recommendations across genres. Also, the engine that we built is not really personal in that it doesn't capture the personal tastes and biases of a user[2]. Anyone querying our engine for recommendations based on a movie will receive the same recommendations for that

movie, regardless of who she/he is. Therefore, in this section, we will use a technique called Collaborative Filtering to make recommendations to Movie Watchers. It is basically of two types:-

- User-Based Filtering : These systems recommend products to a user that similar users have liked. For measuring the similarity between two users we can either use pearson correlation or cosine similarity. Although computing user-based CF is very simple, it suffers from several problems. One main issue is that users' preference can change over time. It indicates that precomputing the matrix based on their neighboring users may lead to bad performance. To tackle this problem, we can apply item-based CF.

- Item-Based Filtering : Instead of measuring the similarity between users, the item-based CF recommends items based on their similarity with the items that the target user rated[3]. Likewise, the similarity can be computed with Pearson Correlation or Cosine Similarity. The major difference is that, with item-based collaborative filtering, we fill in the blank vertically, as oppose to the horizontal manner that user-based CF does. The following table (Fig 3.14) shows how to do so for the movie Me Before You :

	The Avengers	Sherlock	Transformers	Matrix	Titanic	Me Before You
A	2		2	4	5	2.94*
B	5		4			1
C			5		2	2.48*
D		1		5		4
E			4			2
F	4	5		1		1.12*
Similarity	-1	-1	0.86	1	1	

Figure 2.10 Item Based Filtering

It successfully avoids the problem posed by dynamic user preference as item-based CF is more static. However, several problems remain for this method. First, the main issue is scalability. The computation grows with both the customer and the product. The worst case complexity is O(mn) with m users and n items. In addition, sparsity is another concern. Take a look at the above table again. Although there is only one user that rated both Matrix and Titanic rated, the similarity between them is 1. In extreme cases, we can have millions of users and the similarity between two fairly different movies could be very high simply because they have similar rank for the only user who ranked them both.

Combined Recommender : In this project, we have basically implemented three different recommendation systems, namely KNN based engine, content-based filtering with text mining and collaborative filtering. In the end, we combined the results of these recommendation engines using a hybrid ensemble approach which uses autoencoders to encode the text data of movies onto a lower dimension, which gives us meaningful relations. Projecting Data into Lower Dimension :

- Since the data of movies is in plain language we need some way to decompose it into some numerical values and project it to a lower dimension so that we get some meaningful data for our model.
- We use TFIDF Vectorizer to convert text data (information of overview, genre, director, etc.) into numerical data which represents how relevant each word in the document is.
- Some more numerical information regarding vote average, budget, revenue, etc. were added.
- The data was normalized using sklearn preprocessing.
- Word embeddings of each feature were generated to provide a dense representation of words and their relative meanings, with the help of Keras Embedding

layers.[4]

- Now we get a set of meaningful normalized data to be used for our ensemble purpose.

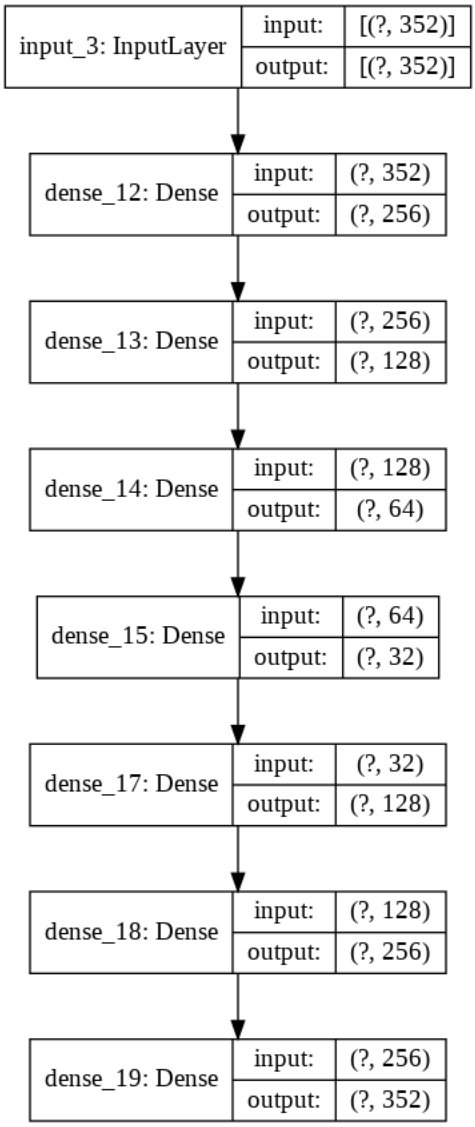


Figure 2.11 Autoencoder Model

Autoencoder model : The autoencoder consists of densely connected layers where the size of layers decrease gradually to the point where we get the encoded data representation projected onto a lower dimension. Then the data is decompressed again with a similar sequence of dense layers with ELU activation[5]. The model is

compiled with Mean Squared Error loss and Adam optimizer. The model learns to regenerate the features of movies quite accurately and we use the encoded information to make a hybrid ensemble which gives us the combined movie recommendation from the different models discussed above.

The Fig 3.12 describes the Model

Ensemble using Encoded Data :

- We need the set of movies predicted by each of the models and the name of the movie for which we are generating the recommendations[6].
- The encoded data vector of each movie is taken.
- Cosine similarity score is used as a distance metric to compute the distance of the recommended movies from the original given movie.
- These are convert it into a list of tuples where the first element is the index and the second is the similarity score.
- The aforementioned list of tuples is sorted based on the similarity scores; that is, the second element.
- Get the top K elements of this list.
- We thus get the titles of the movies based on the final combined movie recommendation.

Fig 3.13 illustrates the Final Movie Recommendation Output.

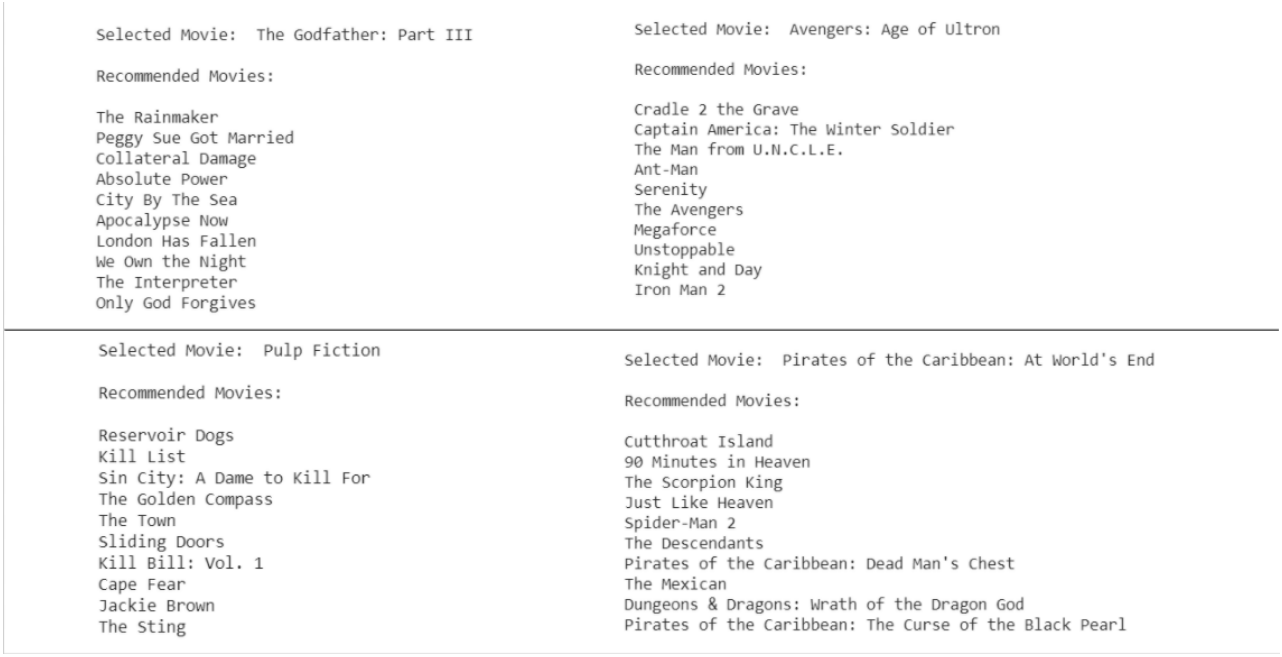


Figure 2.12 Final Recommended Movies

Chapter 3

Conclusions and Discussion

Recommendation Systems in the world of machine learning have become very popular and are a huge advantage to tech giants like Netflix, Amazon and many more to target their content to a specific audience. These recommendation engines are so strong in their predictions that they can dynamically alter the state of what the user sees on their page based on the user's interaction with the app. In this project, we have basically implemented three different recommendation systems, namely KNN based engine, content-based filtering with text mining and collaborative filtering. In the end, we combined the results of these recommendation engines using a hybrid ensemble approach which uses autoencoders to encode the text data of movies onto a lower dimension, which gives us meaningful relations. This helps us to build a robust recommendation system taking the best features of all kinds of models.

Bibliography

[1] Cui, Bei-Bei, "Design and implementation of movie recommendation system based on knn collaborative filtering algorithm," *ITM Web Conf.*, vol. 12, p. 04008, 2017. [Online]. Available: https://doi.org/10.1051/itmconf/20171204008

[2] M.-H. Chen, C.-H. Teng, and P.-C. Chang, "Applying artificial immune systems to collaborative filtering for movie recommendation," *Advanced Engineering Informatics*, vol. 29, no. 4, pp. 830 – 839, 2015, collective Intelligence Modeling, Analysis, and Synthesis for Innovative Engineering Decision Making Special Issue of the 1st International Conference on Civil and Building Engineering Informatics. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1474034615000518

[3] P. Pirasteh, J. J. Jung, and D. Hwang, "Item-based collaborative filtering with attribute correlation: A case study on movie recommendation," in *Intelligent Information and Database Systems*, N. T. Nguyen, B. Attachoo, B. Trawiński, and K. Somboonviwat, Eds. Cham: Springer International Publishing, 2014, pp. 245–252.

[4] C. CHRISTAKOU, S. VRETTOS, and A. STAFYLOPATIS, "A hybrid movie recommender system based on neural networks," *International Journal on Artificial Intelligence Tools*, vol. 16, no. 05, pp. 771–792, 2007. [Online]. Available: https://doi.org/10.1142/S0218213007003540

[5] B. Yi, X. Shen, Z. Zhang, J. Shu, and H. Liu, "Expanded autoencoder recommendation framework and its application in movie recommendation," in *2016 10th International Conference on Software, Knowledge, Information Management Applications (SKIMA)*, 2016, pp. 298–303.

[6] J. Zhao, X. Geng, J. Zhou, Q. Sun, Y. Xiao, Z. Zhang, and Z. Fu, "Attribute mapping and autoencoder neural network based matrix factorization initialization for recommendation systems," *Knowledge-Based Systems*, vol. 166, pp. 132 – 139, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0950705118306178

Appendix A

Below contains links of our Project(To view Click on them):

1. Project Code

2. Dataset