Symbiosis Centre for Information Technology

(A constituent Institute of Symbiosis International (Deemed) University, estd. Under Section 3 of UGC Act 1956)

MBA ITBM Batch 2016-18

FINAL PROGRESS REPORT

A STUDY ON RECOMMENDATION SYSTEM TO BUILD MOVIE RECOMMENDER

By
16030241070
Keerti Chouhan
MBA-ITBM Batch 2016-18

Symbiosis Centre for Information Technology

(A constituent Institute of Symbiosis International (Deemed) University, estd. Under Section 3 of UGC Act 1956)

MBA ITBM Batch 2016-18

ACKNOWLEDGEMENT

I would like to take this opportunity to thank SCIT for providing me an opportunity to work intensely in this dissertation project as part of the curriculum. This, undoubtedly, is a very significant and encouraging aspect about the whole curriculum. The learning in this period has helped me enhance my knowledge about the domain. It would not have been possible to complete this project without the support, guidance and patience of the following people. It is to them; I owe my deepest gratitude.

I would like to thank my mentor Dr. Dhanya Pramod, Director, Symbiosis Centre for Information Technology, Pune, for her invaluable inputs and encouragement throughout the duration of internship. I am highly grateful to her for his guidance and constant support as well as for providing necessary information regarding the project which helped me to learn multiple things in a short span.

Keerti Chouhan
MBA (ITBM),2016-18

Contents

Abstract

On the Internet, where the quantity of decisions is overpowering, there is the need to channel, organize and effectively convey applicable data keeping in mind the end goal to lighten the issue of data over-burden, which has made a potential issue to numerous Internet clients. Recommender frameworks take care of this issue via seeking through vast volume of powerfully created data to give clients with customized substance and administrations. This paper investigates the distinctive attributes furthermore, possibilities of various forecast strategies in suggestion frameworks with a specific end goal to fill in as a compass for research and practice in the field of proposal frameworks. The research would be more concentrated towards building a movie recommendation on python using its various libraries.

CHAPTER 1

1.1 Introduction

A large catalogue of data is available in digital space for any object. The need for recommendation systems came into being because we have moved from an era of scarcity to an era of abundance. To ease tasks of users to find a particular item which suits their prospective demand/requirement from abundant items has given rise to the science of recommendation systems. For this reason, its understanding is required as digital world is expanding every single day and recommendation has become major marketing business to lure customers.

The crux of good recommendation system lies in how is it evaluated. Earlier the prediction power is only used for predicting a user’s choice. However, it is not sufficient for accurate predictions. It is now widely united that correct predictions are crucial, however scarce to deploy a good recommendation engine is. Many times applications’ target is not just to give exact anticipation but to recommend in such a way that users discover new things. For this different properties which may influence the evaluation application wise is considered. Then how a recommender system performs can be evaluated in a proper way.

The main purpose of this study is to learn, analyze recommender systems, its types and build a movie recommender system which will help in recommending users with better movie options. This research explores the potential of different recommendation techniques like content filtering, collaborative filtering, hybrid filtering, their pros and cons. How to overcome different problems like cold start, sparsity, transparency, overspecialization, trust etc. which come across while building different models is described. The different methods available for filtering through these different techniques are studied and mentioned like matrix factorization, word2vec, etc.

1.2 Scope

A movie recommender model is to be built using numpy, pandas, seaborn, matplotlib, scipy, sklearn, nltk, surprise etc. libraries in python etc. This would be checked by different user satisfaction criteria through statistical and decisional evaluation approaches. The data collected for the purpose is MovieLens from GroupLens Research Project at the University of Minnesota. It contains 100k movie ratings from 943 users and a selection of 1682 movies. The work is limited to dataset which has 100,000 ratings (1-5) from 943 users on 1682 movies where each user has rated at least 20 movies. It also has simple demographic info for the users (age, gender, occupation, zip) and genre information of movies. The expected result is learning and understanding of this system and achieving practicality.

1.3 Objective

The objectives are:

- To study recommendation systems, their types and their working.
- To study evaluation criteria, problems arising with filtering techniques/error measures.
- To analyze and build movie recommendation system with the MoviLens data set.

1.4 Methodology

Main methodologies to be followed are:

- Data exploration and cleaning- grouping by roots, imputing missing ness.
- Descriptive data analysis, learning similarity and popularity.
- Model based collaborative filtering using singular value decomposition etc.
- Memory based collaborative filtering by cosine distance calculation.
- Content based filtering using simple text processing methods.

CHAPTER 2

2.1 Review of Literature

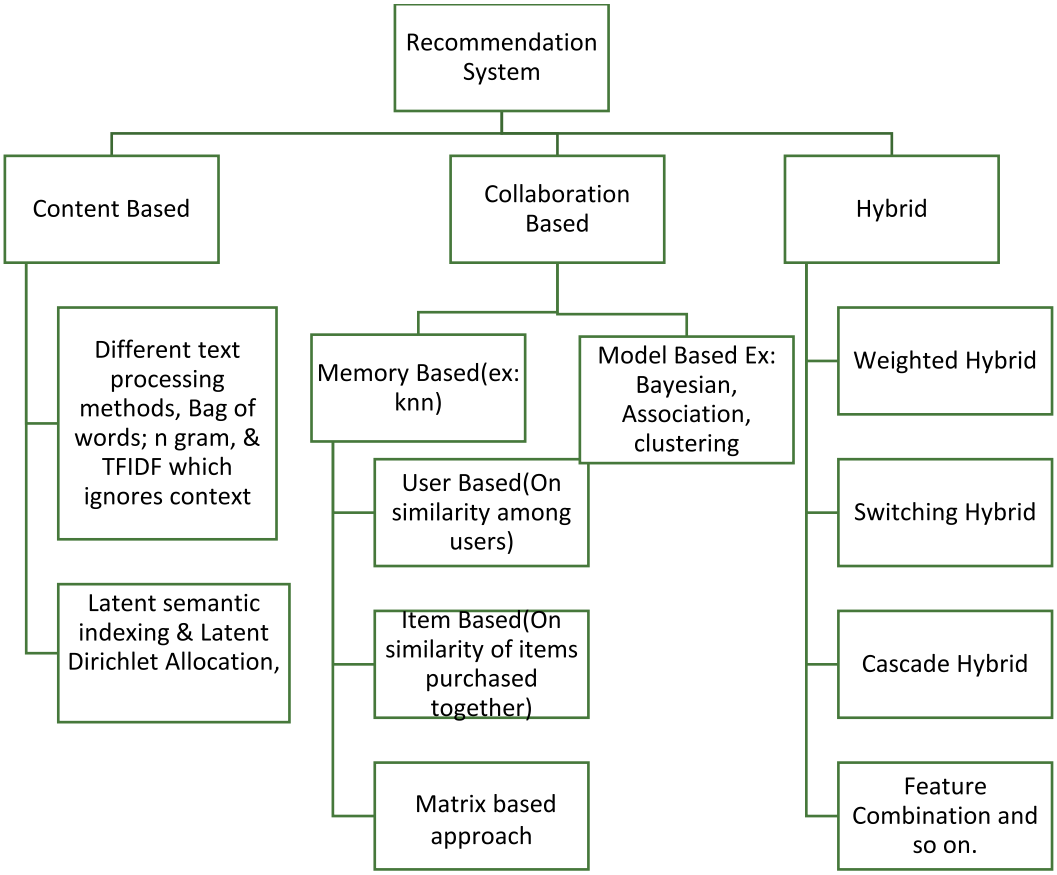
Recommendation systems are information filtering techniques which helps in data prioritization by providing different users with personalized recommendations which is either based on users' behavior, items' behavior or both. At previous times information retrieval systems like Google, DevilFinder have not focused on recommendations as data was not huge until internet became a daily life after .com burst.

The huge availability of data on internet makes it important to display only relevant information to its users. For example, a user searches for shoes on an online shopping platform. Based on a user's behavior or type of shoes a user is interested in, that e shopping can give a personalized suggestion to that user. This requires contextual information which can be location, time, weather, companion or some other situation of behavior [5]. This personalized suggestion can even enhance revenues as seller with better recommendation system can sell more [9].

How recommendation systems have added to in the economy of some of the famous websites is given below [16].

Netflix	2/3rd of the movies watched are recommended
Google News	recommendations generate 38% more click-troughs
Amazon	35% sales from recommendations
Choicestream	28% of the people would buy more music if they found what they liked

There are 3 broad categories of building a recommendation system. They are collaborative filtering, content filtering and hybrid filtering. Their use is based on type of business and requirement of business.


Recommendation System
Content Based
Collaboration Based
Hybrid
Different text processing methods, Bag of words; n gram, & TFIDF which ignores context
Latent semantic indexing & Latent Dirichlet Allocation,
Memory Based(ex: knn)
Model Based Ex: Bayesian, Association, clustering
User Based(On similarity among users)
Item Based(On similarity of items purchased together)
Matrix based approach
Weighted Hybrid
Switching Hybrid
Cascade Hybrid
Feature Combination and so on.

Collaborative filtering method uses users' data. It identifies similar users and recommends them with each other's taste. For example, suppose user A is same as user B and user C in terms of movies they prefer and watch. If user B and user C have watched a particular movie but user A has not, then it will recommend the same user A as their taste in movies are same [8]. In collaborative filtering, a small group of similar customers is found, and a list of recommended items is created, comprised of items that the user is most likely to select. There are two major approaches to collaborative filtering, user-based and item-based. In the user-based approach, the list of recommended items is created based on the customers, while in the item-based approach the list is created based on the products [12]. Collaborative filtering is used by Amazon, Spotify, Facebook, LinkedIn, Google News, MySpace etc. Cold start, sparsity and scalability are the issues with this filtering. Further trust issue arises with the evaluation of certain users whose data is less [4] [8].

The two most popular similarity measures are correlation-based and cosine-based. Pearson correlation coefficient is used to measure the extent to which two variables linearly relate with each other. Cosine similarity is different from Pearson-based measure in that it is a vector-space model which is based on linear algebra rather that statistical approach. It measures the similarity between two n-dimensional vectors based on the angle between them [15]. Cosine-based measure is widely used in the fields of information retrieval and texts mining to compare two text documents, in this case, documents are represented as vectors of items/users [1].

Matrix Factorization is also a famous algorithm which removes the co-clustering problem for collaborative recommendation system. Each item is characterized by a vector of attributes or features inferred from ratings [7]. Word2Vec vector output can also be input to matrix factorization for recommendation. This how mix of of these two can result in a hybrid model.

A major advantage of matrix factorization is that sparsity issue can be resolved using this model. A typical model associates each user u with a user-factor vector, and each movie v with a movie-factor vector. The prediction is then given by [7]
Predicted rating= Baseline estimate + (Transpose of User Factor Vector * Movie Factor Vector)
Where Baseline estimate= estimate for unknown movie rating= average rating + movie bias + user bias

Content filtering method uses item's data. For example, user A watches movie with genre as drama from Salman Khan's production house always then if there is any movie of the same genre and same production house then it would be suggested to user A. Content filtering is used by Pandora, IMDB, Rotten Tomatoes, Jinni, Movie Lens, Rovi Corporation etc. The problem arises when content is limited leading to sparsity of data. With sparse content quality of recommendation becomes low. Also it suggests items which are already consumed which is called over specialization. Simple approaches use the average values of the rated item vector while other sophisticated methods use machine learning techniques such as Bayesian Classifiers, cluster analysis, decision trees, and artificial neural networks in order to estimate the probability that the user is going to like the item [17].

When we have an idea about which content is to be considered for a particular item, then data needs to be transferred to a vector space model which is also an algebraic representation of text/ unstructured data. There are two approaches for it. First is bag of word model, which considers term frequency and inverse document frequency. It tells how important a word is in a document and in a corpus. However, this approach does not consider the context of words. So when context is to be considered Latent Dirichlet Allocation and Latent Semantic Indexing is to be used [20]. In both of these algorithms we try to find a latent variable from other given variables. Dirichlet is a distribution type which is assumed in text. These algorithms discover on its on what is the topic of documents etc., by decomposing document term matrix into document topic and topic term matrix [13] [14] [20].

From some of the common methods used for recommendation are neighborhood based, machine learning based, matrix factorization based (for collaboration) etc. From natural language processing domain, it is word2vec (for content) for recommendation [6].

Word2Vec is a neural network class which takes in input a large corpus of word and generates a vector of numbers as output based on either skip gram model or bag of words model. As bag of words approach misses out context, word2vec for recommendation focuses on skip gram model. The vector of numbers in output can be used in recommendations as well. In recommendation system list of movies which users have seen can be taken as input and using the movies they have watched before and after can be used to teach our model that those movies somehow belong to the same context [6].

Further, word2vec can be used in movie recommendation can be summarized in below 4 points:
1. All the movies seen by users are located first.
2. Scores of these movies are averaged to get a user vector,
3. Search for the nearby movies,
4. Keep the nearby movies to user vector as recommendation.

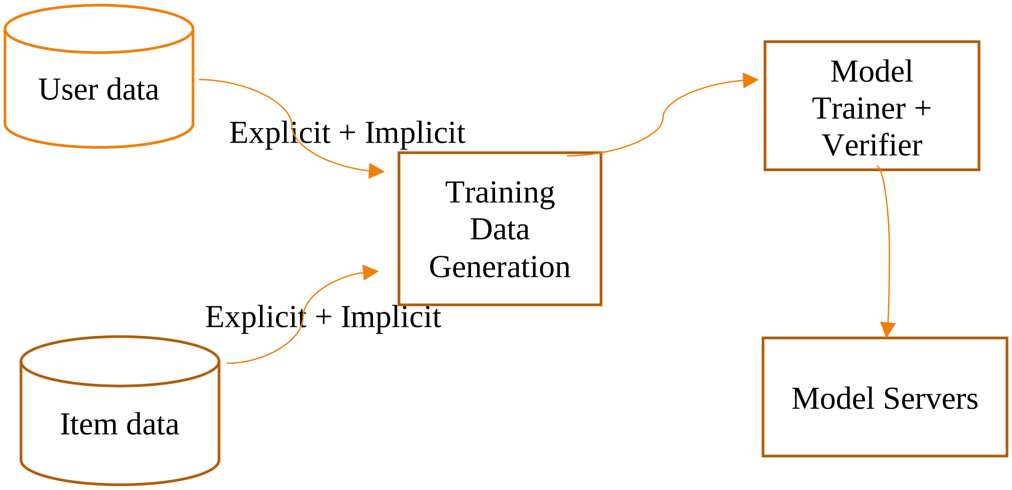
The difference between content based filtering and item based collaborative filtering is content based filtering suggests items based on similarity of items rated for example, a user taking leather jackets online from myntra will be suggested the same for new stocks. While item based collaborative filtering suggests items which are bought together by similar kind of users for example, user A and user B display similar characteristics and user A purchases tube light and fan together so user B will also be suggested the same.

Hybrid Filtering method actually combines content filtering and collaborative filtering to get the better results, accuracy and performance. Starting with content filtering on which user watches which type of movies and feeding this data to collaboration filter can give a better result [11]. Suppose user A and user B have developed a new taste in adventure genre then seeing the type of content user A, user B and user C are watching and seeing the similarity is rating that they are giving, user C can be recommended with those adventure genre movies which are liked by user A and user B. Netflix uses hybrid filtering method for movie recommendation.
The problem which can arise with this filtering is scalability issue. For this systems should be made scalable such that more physical assets can be added or removed as desired [4].

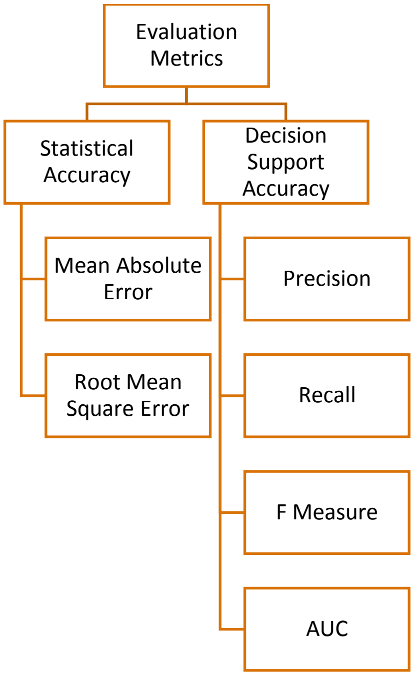
The weighted recommendation takes input from various recommendation systems and calculates the weighted average through linear approach. Weights can be equal or unequal or even adjusted as per predictions. Switching hybridization method switches from one method to another based on the data to produce a decent rating. Its only drawback is it makes the system more complex. Cascade hybridization works on taking output of one system and giving input to the other. It is based on continuous refinement process. In feature hybridization technique takes input features from other recommendations' outputs. For example, the rating of similar users which is a feature of collaborative filtering is used in a case-based reasoning recommendation technique as one of the features to determine the similarity between items [1].

The phases of recommendation include information collection, learning phase and prediction/recommendation phase [1]. In information collection phase users' information, their behavior etc. is collected from their profiles to generate a model for recommendation. There is explicit feedback and implicit feedback that is being collected for information. Explicit feedback is the one which asks users for their ratings explicitly and that's it is more reliable whereas implicit feedback assumes rating based on users' behavior. Then comes the learning phase where a model learns from the data and predicts based on the data. The similarity among users/items for recommendation is found using Pearson Correlations Coefficient.

Data Generation and modelling pipeline [3]



Evaluation of recommendation system is important as different applications suits with different type of recommendations. Different metrics can be used like accuracy and coverage. Accuracy is the fraction of correct recommendation out of total possible recommendations to users as per their need [10]. For example, one user may want to get first edition on Ramayana while other may want to get the latest edition available online. So how good a recommendation system does in this is known by its accuracy. For a recommendation engine Coverage is the fraction of object in the search space the system is able to provide recommendations for. So more the coverage, move would be the accuracy [18].



Where,

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n}|y_j - \hat{y}_j| \quad \mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

$$PRE = \frac{TP}{TP+FP}$$

$$REC = TPR = \frac{TP}{P} = \frac{TP}{FN+TP}$$

$$F_1 = 2 \cdot \frac{PRE \cdot REC}{PRE + REC}$$

Where, n = total no. of predictions, Yj= actual value, Yj^= predicted values, Pre= precision, Rec= recall, TP= true positive of a cross table, FP= false positive of a cross table, FN= false negative of a cross table.

The statistical accuracy prediction tells whether a recommender can accurately recommend the fact that a user likes something whereas decision support accuracy tells whether or not recommender recommends what a user likes.

However, there are many factors apart from accuracy which are considered for user satisfaction in recommendation system. Some of them are data security, item’s lifespan, time to first recommendation and interoperability [2]. Even satisfaction of recommender provider is also considered and many a times a user’s satisfaction may not lead to his provider’s satisfaction. For example, Amazon may want to recommend fascinating electronics items which are high priced so that they can generate more revenue by garbing users' attention. This may misguide users making them unhappy. So a good recommender should also maintain a balance between user's satisfaction and provider's satisfaction [3].

CHAPTER 3

3.1 Analysis of work done

Analysis of problem under research

Problem: The movieLens dataset is used for the purpose of recommender systems which aim to predict user movie ratings based on other users' ratings and content/metadata of movies.
A few questions which I tried solving are:

- The main purpose is to find and suggest movies which users like.
- To know which genres are receiving the highest ratings.
- To know how is genre liking changing over time and so on?

Dataset: The dataset describes 5-star rating and free-text tagging activity from MovieLens, a movie recommendation service. It contains 100004 ratings and 1296 tag applications across 9125 movies. This dataset was generated on October 17, 2016. Users were selected at random for inclusion. All selected users had rated at least 20 movies. No demographic information is included. Each user is represented by an id, and no other information is provided.

This dataset consists of the following files:

a) movie.csv has movie information:

movieId
title
genres

where,
Genres are one or more from the following: Action, Adventure, Animation, Children's, Comedy, Crime, Documentary, Drama, Fantasy, Film-Noir, Horror, Musical, Mystery, Romance, Sci-Fi, Thriller, War, Western, No genre.

b) rating.csv has ratings of movies by users:

userId
movieId
rating
timestamp

c) link.csv has identifiers that can be used to link to other sources:

movieId
imdbId

tmdbId

where, imdb stands for Internet movie database and tmdb stands for the movie database.

d) tag.csv has tags applied to movies by users:

userId
movieId
tag
timestamp

e) genome_tags.csv has tag descriptions:

tagId
tag

f) movies metadata

Understanding movie data through its metadata
adult
belongs_to_collection
budget
genres
homepage
id
imdb_id
original_language
original_title
overview
popularity
poster_path
production_companies
production_countries
release_date
revenue
runtime
spoken_languages
status
tagline
title
video
vote_average
vote_count

Analysis Steps

Data cleaning which involves cleaning of the keywords, finding correlation between them, dealing with missing values. Data exploration by movie keywords, filling missing values, finding no. of movies every year and their genres.

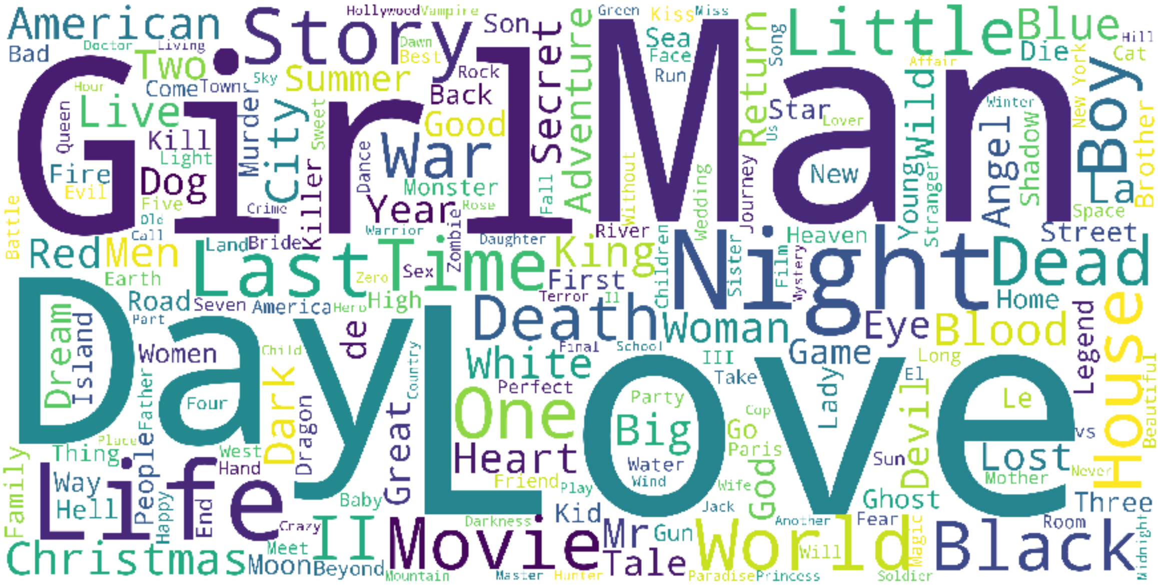
Data cleaning started with removing of such features which are not considered as important for the study and research. Like *original title* of the movie as *title* feature is there for the same. *Revenue* and *budget*, are coerced with nan where 0 is recorded as 0 cannot be any films' budget or revenue. Nan tells that information is not available for that particular movie.

Feature engineering is done to know return on a movie. *Revenue/Budget* gives *return* of a movie. Also, predicted if a movie will be successful or not based on its return. First revenue is predicted using repressor on genre wise revenue and then return is found. If the division is more than 1 than it is profit otherwise loss. This indirectly tells about success of a movie.

Exploratory data analysis

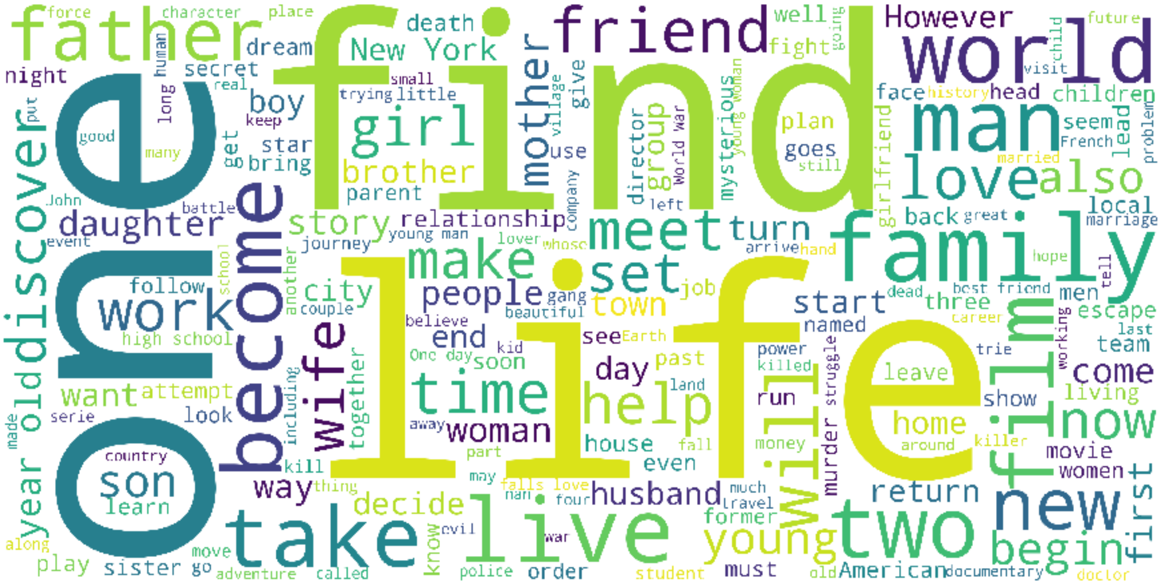
Word cloud for movie *titles* and *overview* are made to know which words are appearing more often and what most movies talk about.

Below is the word cloud of titles:

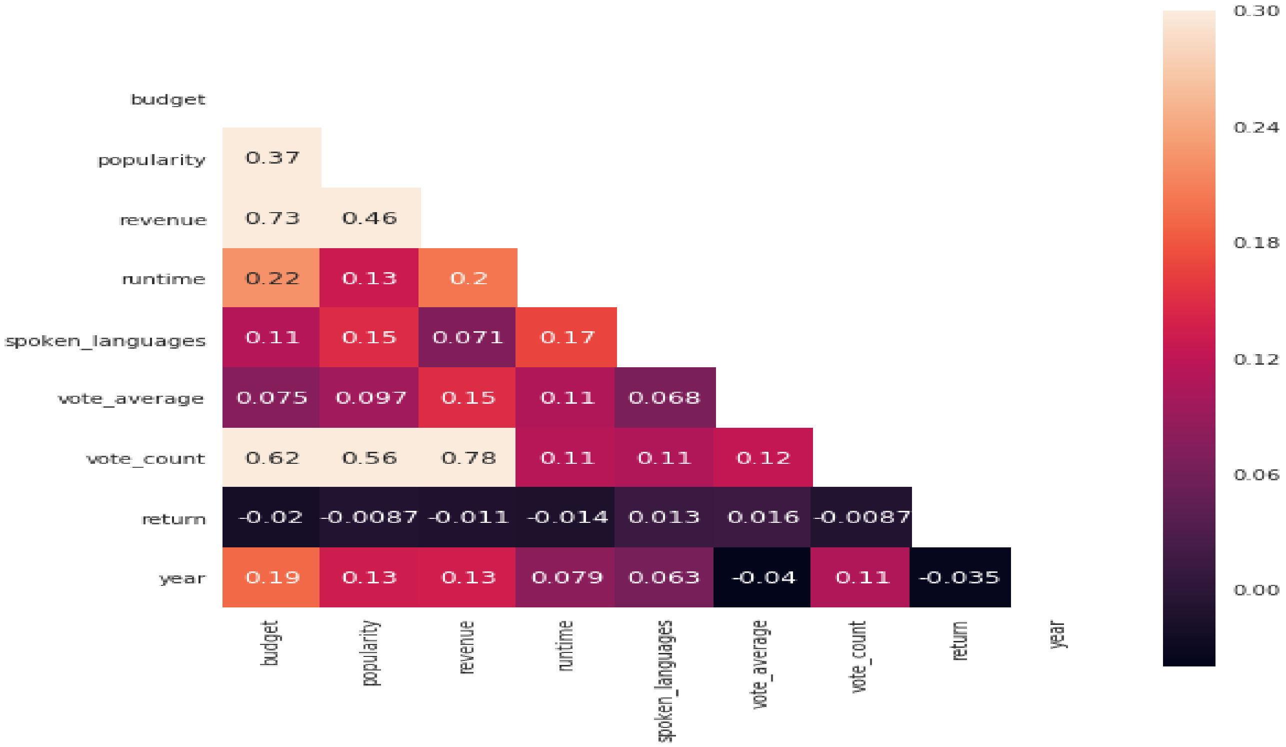


Love, Life, girl, man, day, night etc. are some commonly occurring words in the in the title. I think most movies are on love and life of people.

Below is the word cloud for movie over views: The image tells the popular movie themes are life, wife, family,work, time, father, love and so on.

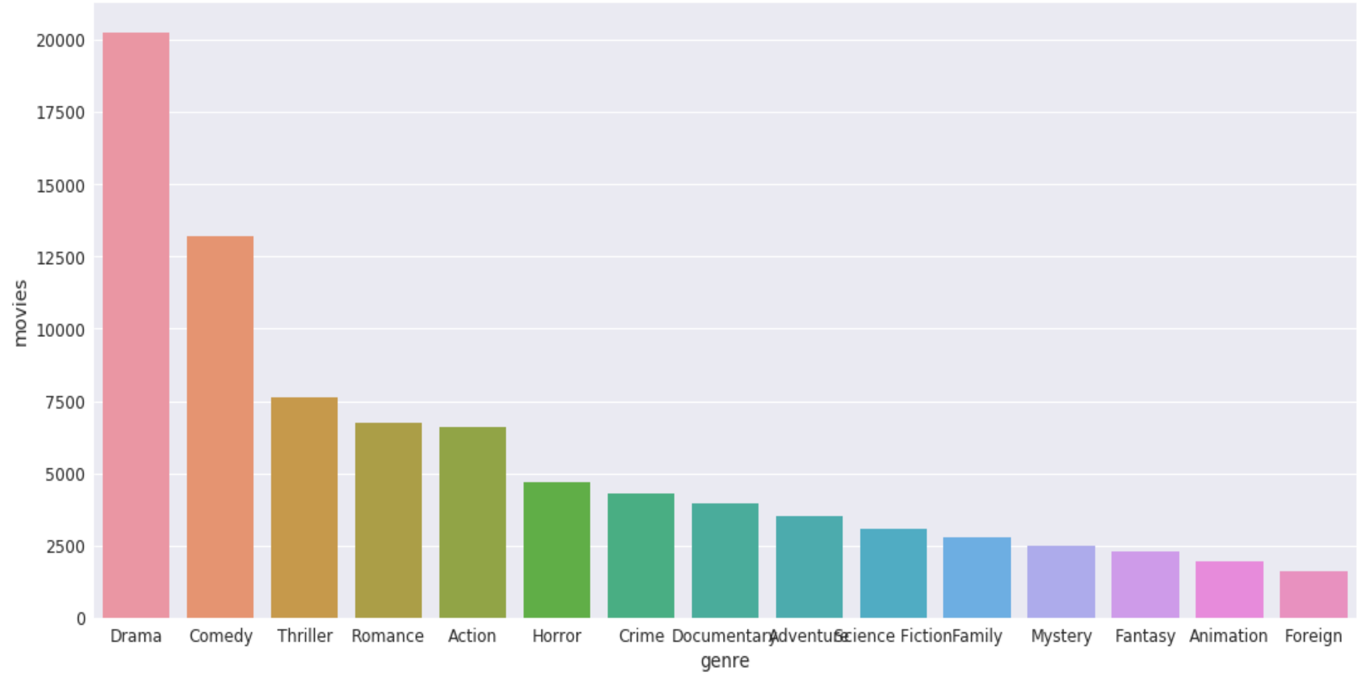


Correlation among metadata variables is found which is as below:

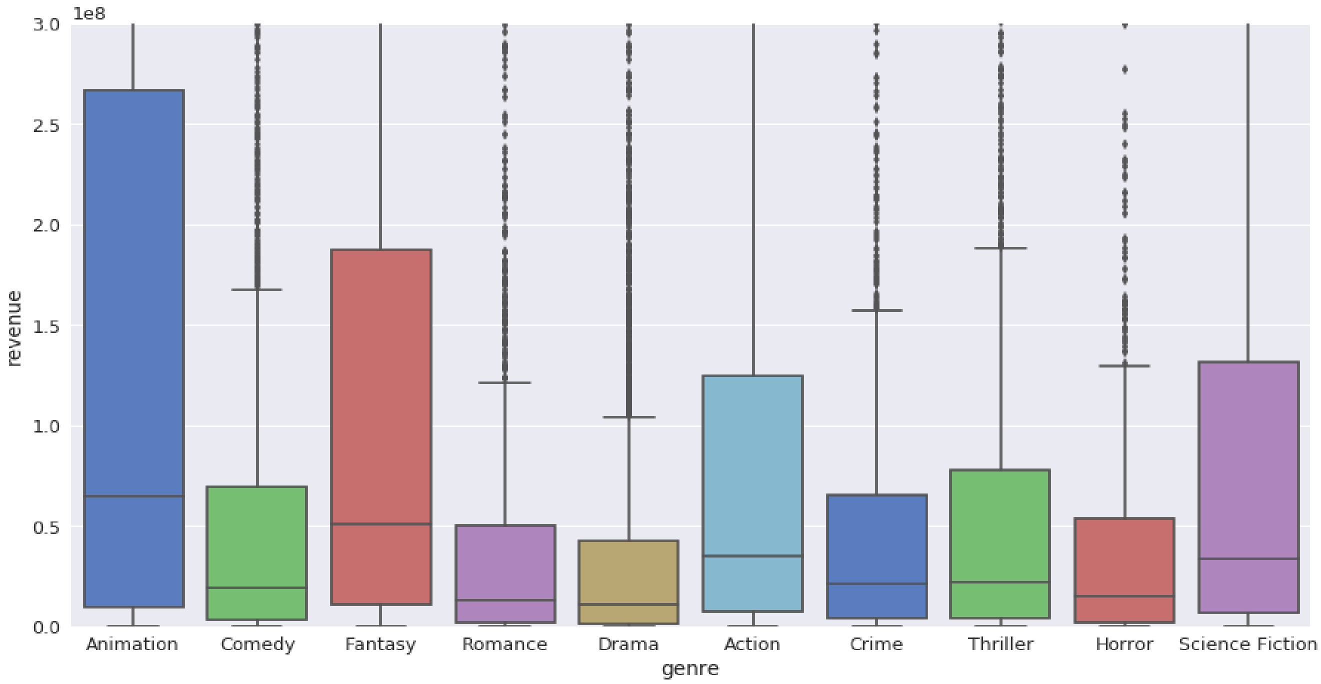


Budget, popularity, vote count and revenue are seen as correlated variables.

Below graph shows the no. of movies released genre wise.



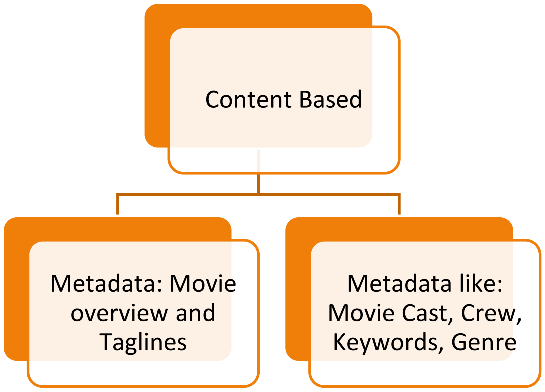
Below box plot graph shows revenue of movies genre wise.



3.2 Proposed Solution

After exploratory data analysis comes the focus comes on the main part of the project, that is, building of movies recommendation system. Below are some solutions for the same.

Content based Recommender: This recommender will be based on similarity among movies based on certain metrics. It will give suggestions for movies based on the movie a particular user has liked. I built two content based engines taking input as follows:

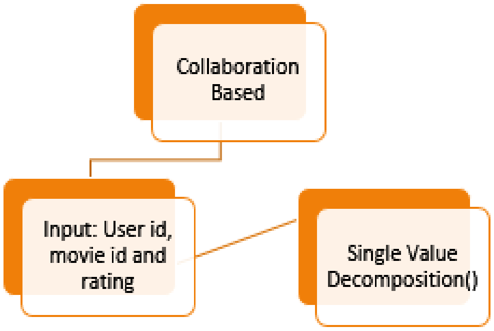

Content Based
Metadata: Movie overview and Taglines
Metadata like: Movie Cast, Crew, Keywords, Genre

Feature engineering is applied on metadata, like only director is picked up and considered a deciding feature as others contribute a little. Algorithm is as below:

1. Then Term frequency inverse document frequency is found which vectored the content.
2. Passed TFIDF matrix to cosine similarity which gives cosine similarity of all the movies.
3. get Recommendation () is built which gives 10 most popular movies based on cosine similarity.

- Input: Title of movie
- Cosine matrix is converted to list.
- Sorted that list.
- For loop to return 30 matching movies based on similar content

Collaborative Recommender: *First way:* I used Surprise library to build this model which used single value decomposition function instead of building this from scratch. The RMSE gotten have been under 1 and the model give estimated rating for a given user id, movie id. SVD function thus does not care what the movie is all about. It purely works on assigned movie id and how other users have given rating and it is irrespective of movie content. SVD.train() and SVD.predict() are two main functions used here.


Collaboration Based
Input: User id, movie id and rating
Single Value Decomposition()

Second Way: Building from scratch
1. To work on userid and movie id from movies ratings data set.
2. Converting those two columns into pivot table with row as user id and column as movie id.
3. Apply matrix factorization to find these latent features(U, M).
4. Find all predicted ratings by multiplying U and M matrices.
5. Similarity of movies is found based on the difference between movie features like subtract the current movie's features from every other movie's features found from latent extraction.
6. Recommendation is given with input as user id, from this we get movie id, join on movies data to get all the movies which are not already reviewed

3.3 Alternative Solutions and their advantages & disadvantages

(Alternates and future scope)

- Hybrid Recommender: This can be brought together starting with content followed by collaboration which offered motion picture suggestions specific to users' dependent on internal rating that was calculated for user. Input can be user id and title of movie. It takes the advantage of both, similarity among users and the content of data and suppresses the drawback of both the recommenders. Disadvantage is just that it will give rise to complexity of operations.

- The language in which the film is made is not being considered as a feature for recommendation. This could be a deciding feature in fact as a cast in Tollywood and Bollywood may be same but user may not want to watch movie in some other language other than preferred one.

- Another purpose worries those supplanting of the keywords by all the more incessant synonyms. Over a portion cases, it might have been indicated that the synonyms chosen meant something different in that context. Definitely, the entire transform may merit more consideration.

- Use of word to vectors which is a shallow neural network to convert content into vectors can be used instead of traditional TFIDF approach. Advantage of word to vec is it takes into account context of words, that is it captures position in text, semantics and co-occurances in different documents while TFIDF does not. TFIDF is good for lexical features but not for capturing semantics which is only possible through topic modellings.

3.4 Technical justification of the solution

Technical justification of content based system:

To understand the content, I started with content based filtering. When the users input is more, this system gives more accurate results as it works on movie information and user profiles.

The main points of building this system are:

- Results are quiet relevant.
- Suggestions are quiet transparent as it's based on movie features unlike collaboration filtering where users may not get to know why are they getting such suggestions.
- New items can be recommended as soon as it is added in the dataset.
- It is technically way easier than implementing collaboration filtering. As the science behind it is straightforward.

Technical justification of collaboration based system:

Our content built system experiences a few extreme limits. It may be best skilled for suggesting pictures which are similar to some pictures. That is, it will be not skilled from claiming and catching tastes and furthermore giving suggestions over genres is a difficult task.

Also, the system that we manufactured will be not truly personal in that it doesn't catch those particular tastes and inclinations of a user. Anybody querying the system for a picture will get those same suggestions to that movie, in any case for who s/he will be. Therefore, collaborative filtering is used to anticipate upon the amount one will like a specific movie item which others have seen but that person has not.

Below is some of the glimpse of code for some recommendation:

```
import numpy as np
import pandas as pd
import matrix_factorization_utilities

df = pd.read_csv('movie_ratings_data_set.csv')

movies_df = pd.read_csv('movies.csv', index_col='movie_id')

ratings_df = pd.pivot_table(df, index='user_id', columns='movie_id', aggfunc=np.max)

U, M = matrix_factorization_utilities.low_rank_matrix_factorization(ratings_df.as_matrix(),
                                                                    num_features=15,
                                                                    regularization_amount=1.0)
M = np.transpose(M)
```

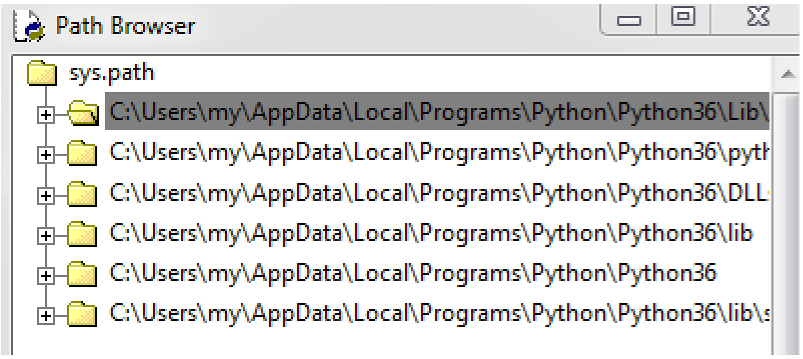
```
# Choose a movie to find similar movies to. Let's find movies similar to movie #5:
movie_id = 5
movie_information = movies_df.loc[movie_id]
print("We are finding movies similar to this movie:")
print("Movie title: {}".format(movie_information.title))
print("Genre: {}".format(movie_information.genre))

current_movie_features = M[movie_id - 1]

print("The attributes for this movie are:")
print(current_movie_features)

# The main logic for finding similar movies:
difference = M - current_movie_features
absolute_difference = np.abs(difference)
total_difference = np.sum(absolute_difference, axis=1)

# Create a new column in the movie list with the difference score for each movie
movies_df['difference_score'] = total_difference
sorted_movie_list = movies_df.sort_values('difference_score')
print("The five most similar movies are:")
print(sorted_movie_list[['title', 'difference_score']][0:5])
predicted_ratings = np.matmul(U, M)
print("Enter a user_id to get recommendations (Between 1 and 100):")
user_id_to_search = int(input())
print("Movies previously reviewed by user_id {}:".format(user_id_to_search))
reviewed_movies_df = raw_dataset_df[raw_dataset_df['user_id'] == user_id_to_search]
reviewed_movies_df = reviewed_movies_df.join(movies_df, on='movie_id')
print(reviewed_movies_df[['title', 'genre', 'value']])
input("Press enter to continue.")
print("Movies we will recommend:")
user_ratings = predicted_ratings[user_id_to_search - 1]
movies_df['rating'] = user_ratings
already_reviewed = reviewed_movies_df['movie_id']
recommended_df = movies_df[movies_df.index.isin(already_reviewed) == False]
recommended_df = recommended_df.sort_values(by=['rating'], ascending=False)
print(recommended_df[['title', 'genre', 'rating']].head(5))
rmse_training =
matrix_factorization_utilities.RMSE(ratings_training_df.as_matrix(),predicted_ratings)
rmse_testing =
matrix_factorization_utilities.RMSE(ratings_testing_df.as_matrix(),predicted_ratings)
print("Training RMSE: {}".format(rmse_training))
print("Testing RMSE: {}".format(rmse_testing))
```

3.5 Technical environment details

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

1. Downloaded python 3.6.4 from the following source as per my systems' specification : https://www.python.org/downloads/windows/

2. Installed python by running the executable file and browsed the default path for saving files. This is the same path which contains python directory.

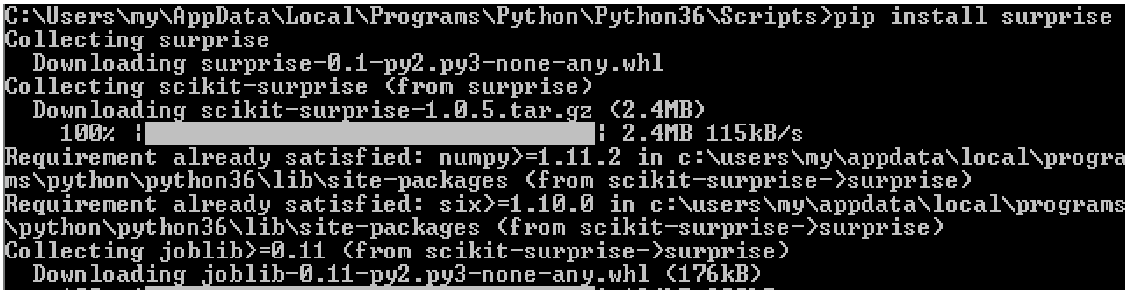


```
sys.path
  C:\Users\my\AppData\Local\Programs\Python\Python36\Lib\
  C:\Users\my\AppData\Local\Programs\Python\Python36\pyth
  C:\Users\my\AppData\Local\Programs\Python\Python36\DLL
  C:\Users\my\AppData\Local\Programs\Python\Python36\lib
  C:\Users\my\AppData\Local\Programs\Python\Python36
  C:\Users\my\AppData\Local\Programs\Python\Python36\lib\
```

3. Some modules are downloaded with the python package index installation by default in python 3 and above versions. To install those modules, following command is to be used as, open command prompt and reach the default path. The default path is changed using change directory command as follows:

   cd C:\Users\my\AppData\Local\Programs\Python\Python36

   Once this path is reached, following command will install the desired modules:
   pip install surprise
   Following is the snapshot telling type of process that keeps running at the back for installation of libraries.



```
C:\Users\my\AppData\Local\Programs\Python\Python36\Scripts>pip install surprise
Collecting surprise
  Downloading surprise-0.1-py2.py3-none-any.whl
Collecting scikit-surprise (from surprise)
  Downloading scikit-surprise-1.0.5.tar.gz (2.4MB)
    100% |████████████████████████████████| 2.4MB 115kB/s
Requirement already satisfied: numpy>=1.11.2 in c:\users\my\appdata\local\progra
ms\python\python36\lib\site-packages (from scikit-surprise->surprise)
Requirement already satisfied: six>=1.10.0 in c:\users\my\appdata\local\programs
\python\python36\lib\site-packages (from scikit-surprise->surprise)
Collecting joblib>=0.11 (from scikit-surprise->surprise)
  Downloading joblib-0.11-py2.py3-none-any.whl (176kB)
```

4. Modules which are not included are to be downloaded from the following source: https://www.lfd.uci.edu/~gohlke/pythonlibs/ . File format downloaded are Wheel files. Copy those from download folder to default python folder and in the command prompt use the the same pip install command with the downloaded file name.

5. Start executing code by importing libraries on python shell.

6. Python help document can be accessed easily by pressing "F1 key" while using python shell.

3.6 Possible applications in the industry

This recommender system can help movie industry in the following ways :

1. Form habits: Serving accurate content can trigger cues, building strong habits and influencing usage patterns in customers.

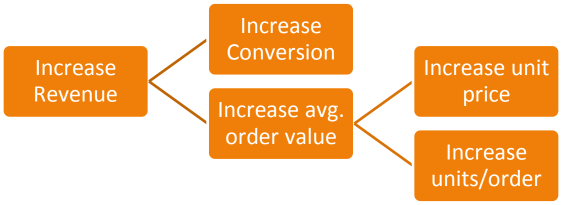


Diagram showing options to increase sale

2. Improve retention: It can improve retention by continuously catering to users as per their preferences which can make them loyal subscriber of the service.

3. Increase Sales: Various research show an increase in upselling ranging from 10-50% is caused by accurate "You might also like product recommendations."

4. Wise decisions: Investors and other partners can take wise decisions. There are various movie genres which are liked by a particular generation at a particular time. How successful their production could be is easy to predict.

5. Cost savings: Hit movie patterns can be studied to make future hit movie and cost of the previous ones can be studied in detail to avoid unwanted overheads.

6. Accelerate work: Analysts can save upto 80% time when served tailored suggetsions for materials necessary for their further research

7. Increased brand loyalty: When a customer feels as though they are understood by your brand, they are more likely to stay loyal and continue purchasing through your site

CHAPTER 4

4.1 Findings

Following are the findings:

1. There are a total of **45,466 movies** with **24 features**.

2. Could find the revenue of a particular movie by regression.

3. Could find if a movie will be hit or flop by classification or by introducing a return variable and analyzing based on its value.

4. Word cloud of titles show ubiquitous presence of some chemistry in the movies.

5. ‘Overview’ wordcloud tells the popular movie themes are life, wife, family,work, time, father, love and so on.

6. Genre wise movies are mostly released in drama, comedy, thriller, romance in the following order.

7. Genre wise revenue is in the following order: animation, comedy, fantasy and so on.

8. How genre liking is changing over time.

9. **Vote Count** is the most important attribute identified by our Classifier. Other important attributes are **Budget**, **Popularity** and **Year**.

10. Movie recommendations based on content and collaborative filtering as explained earlier.

4.2 Recommendations

1. **Content Based Recommender:** I have built two content based engines; the one which takes input as movie overview and taglines and the other which takes input as metadata such as cast, crew, genre and keywords to come up with predictions. Main logic behind processing was simple natural language processing by getting term frequency, document frequency and cosine similarity.
2. **Collaborative Recommender:** Further I have built two content based recommender where in the first one I have made use of Surprise Library of python to build a collaborative filter based on single value decomposition. The root mean square error obtained was less than 1 and the engine gave estimated ratings for a given user and movie. The other was built on the concept of matrix factorization and obtaining latent variables from it. Similarities of movies is found based on the differences in movies features.

3. **Hybrid Engine**: Further work is to bring together ideas from content and collaborative filtering to make a model which would give movie suggestions to a particular user based on the estimated ratings that it had internally calculated for that user.

4.3 Conclusion

Recommender frameworks open new chances for retrieving and customizes majority of the data on the web. It likewise serves with allay to those issues of majority of the data over-burden which will be a regular burden for data recovery frameworks and empowers users need to get benefits which are not promptly accessible to them. Firstly, this paper examined the two customary suggestion system i.e. collaboration filtering and content filtering, what's more highlighted is their qualities, different taking in calculations utilized within generating suggestion models and assessment measurements utilized within measuring the nature.

With this research work I got an understanding of how recommendation systems work and what evaluation criteria are to be considered. This study has implemented recommendation systems on movies dataset. The system has considered pros and cons of content and collaborative filtering. Based on users need, the system is able to recommend movies based on various inputs. Matrix factorization, latent factors, product similarity, user defined function for recommendation are some of the middle steps performed for getting recommendations. The system is also evaluated by finding out root mean square error for it. It started with reading research papers on recommendation and movie recommenders intensively. Once the understanding is achieved, movie lens data is exported from the source and data cleaning is done followed by explorative data analysis, descriptive data analysis and predictive data modelling.

The further work which can be performed on it is building hybrid recommender which would be the combination of both the types of engines built, which would consider context of the data and compress drawbacks of both the engines. It will also enhance the execution of the system. Our approach can be further stretched out with different domains to suggest songs, video, venue, news, books, tourism and e-commerce sites, and so forth throughout this way, observing and stock arrangement of all instrumentation may be enhanced.

The environment it is built on is Python 3.6.4 version. The possible applications for this engine are it would help in profit maximization by recommending right movie to the right person. It would also help in cost cuttings as movie makers can learn movie liking patterns and make movies as per the watchers most interest. This system can also prove good for customer retention and thus building brand image. Thus it would serve the purpose of watchers and business people both.

References

[1] F. Isinkaye, Y. Folajimi and B. Ojokoh, "Recommendation systems: Principles, methods and evaluation", *Egyptian Informatics Journal*, vol. 16, no. 3, pp. 261-273, 2015

[2] J. Beel, B. Gipp, S. Langer and C. Breitinger, "Research-paper recommender systems: a literature survey", International Journal on Digital Libraries, vol. 17, no. 4, pp. 305-338, 2015.

[3] H. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, R. Anil, Z. Haque, L. Hong, V. Jain, X. Liu and H. Shah, "Wide & Deep Learning for Recommender Systems", *Arxiv.org*, 2017.

[4] "Survey Paper on Web Recommendation System", International Journal of Science and Research (IJSR), vol. 4, no. 11, pp. 2389-2391, 2015.

[5] S. Wu, Q. Liu, L. Wang and T. Tan, "Contextual Operation for Recommender Systems", IEEE Transactions on Knowledge and Data Engineering, vol. 28, no. 8, pp. 2000-2012, 2016.

[6] From Word Embeddings to Item Recommendation. Turkey: Makbule Gulcin Ozsoy, 2016.

[7] "Matrix Factorization Based Query Recommendation", International Journal of Science and Research (IJSR), vol. 4, no. 12, pp. 169-173, 2015.

[8] R. Katarya and O. Verma, "An effective collaborative movie recommender system with cuckoo search", *Egyptian Informatics Journal*, vol. 18, no. 2, pp. 105-112, 2017.

[9] N. Liu, L. He and M. Zhao, "Social temporal collaborative ranking for context aware movie recommendation", ACM Transactions on Intelligent Systems and Technology, vol. 4, no. 1, pp. 1-26, 2013.

[10] Y. Ishida, T. Uchiya and I. Takumi, "Design and evaluation of a movie recommendation system showing a review for evoking interested", International Journal of Web Information Systems, vol. 13, no. 1, pp. 72-84, 2017.

[11] E. Çano and M. Morisio, "Hybrid recommender systems: A systematic literature review", Intelligent Data Analysis, vol. 21, no. 6, pp. 1487-1524, 2017.

[12] J. Jooa, S. Bangb and G. Parka, "Implementation of a Recommendation System Using Association Rules and Collaborative Filtering", *Procedia Computer Science*, vol. 91, pp. 944-952, 2016.

[13] Y. Ng, "MovRec: a personalized movie recommendation system for children based on online movie features", *International Journal of Web Information Systems*, vol. 13, no. 4, pp. 445-470, 2017.

[14] P. Li, X. Pan and H. Chen, "User Clustering Topic Recommendation Algorithm Based on Two Phase in the Social Network", *International Journal of Multimedia and Ubiquitous Engineering*, vol. 10, no. 10, pp. 233-246, 2015.

[15] "Recommendation system Based On Cosine Similarity Algorithm", International Journal of Recent Trends in Engineering and Research, vol. 3, no. 9, pp. 6-10, 2017.

[16] M. Kumar, D. Yadav, A. Singh and V. Kr., "A Movie Recommender System: MOVREC", International Journal of Computer Applications, vol. 124, no. 3, pp. 7-11, 2015.

[17] Blanda, Stephanie "Online Recommender Systems – How Does a Website Know What I Want?". American Mathematical Society. Retrieved October 31,2016.

[18] F. Hernández del Olmo and E. Gaudioso, "Evaluation of recommender systems: A new approach", *Expert Systems with Applications*, vol. 35, no. 3, pp. 790-804, 2008.

[19] Optimize Recommendation System with Topic Modeling and Clustering, The Fourteenth IEEE International Conference on e-Business Engineering, 2017.

[20] Omar Abdillah, Mirna Adriani "Mining User Interests through Internet Review Forum for Building Recommendation System", 2015 29th International Conference on Advanced Information Networking and Applications Workshops.