Rochester Institute of Technology
B. Thomas Golisano College
of
Computing and Information Sciences

Master of Science in Information Sciences and Technology

~ Project Proposal Approval Form ~

Student Name: Lekha Nahar

Project Title: Evaluating Features, Building recommender system for Amazon TV and movies reviews

Project Area(s):	Application Dev.	Database	Website Dev.
(Ö primary area)	Game Design	HCI	eLearning
	Networking	Project Mngt.	Software Dev.
	Multimedia	System Admin.	Informatics
	Geospatial	Other **Data Analytics**	

~ MS Project Committee ~

Name	Signature	Date
Prof. Erik Golen Chair		
Prof. John-Paul Takats Committee Member		

Extracting Features, building Recommender system for Amazon Movies and TV reviews

By: Lekha Nahar

Project submitted in partial fulfillment of the requirements for the degree of Master of Science in **Information Sciences and Technologies**

Rochester Institute of Technology

B. Thomas Golisano College
of
Computing and Information Sciences

Department of Information Sciences and Technologies

Table of Contents

Contents

Extracting Features, building recommender system for Amazon Movies and TV reviews

Abstract

This project deals with the Amazon dataset provided by Julian McAuley at the University of California San Diego. The project aims to extract the features that are discussed in the Amazon textual reviews using Latent Dirichlet Allocation(LDA). Furthermore, once the features are extracted, a recommender is built. To accomplish this, the project proposes various models such as Topic-clustering Recommendation, Unconstrained Matrix Factorization, and Content-based filtering. Initially, the dataset is cleaned and data exploration is performed to observe the various trends in data. Based on the ratings of the reviews, the word cloud is created to determine the importance of each word in the dataset. After the initial data exploration, topics discussed in the dataset are extracted using Latent Dirichlet Allocation(LDA).[8,10] Finally, using these topics, the recommender is built with the help of different models like Topic-clustering Recommendation, Unconstrained Matrix Factorization, and Content-based filtering. Based on the metrics like Recall and Mean Absolute Error, the best model will be selected.

Keywords: Amazon, Recommender, LDA, Topic Modelling, Content-based Filtering, Matrix Factorization

1. Introduction

The internet is an important source of information. There has been a considerable amount of growth in the field of e-commerce over the past few years. Almost everything that we need is easily available online. Websites like Amazon, eBay, and Flipkart play a vital role when it comes to e-commerce. Over 60% of the population in Asia, Africa/Middle East, and Latin America regions are willing to shop online [7]. It was observed that in the first quarter of 2017, the sales for eCommerce reached US$105.7 billion [10]. If the majority of the population depends on e-commerce websites for shopping it is important to give a gist of reviews that are posted regarding the products on the website. Various other customers read reviews about any product that is posted online. Based on the reviews present and the number of reviews available customers tend to decide whether to buy the product or not. The reviews of any product present on their website play a very important role in deciding the success of either website or product.

The adequate number of competing products and overloaded information may make it difficult for online customers to make choices. One of the disadvantages of such reviews is that they are long sentences which can sometimes be quite difficult to read and understand the gist of the whole sentence [9]. Hence there must be some way to extract the feature of the reviews and display it. Topic modeling is a type of statistical model for discovering the abstract "topics" that occur in a collection of documents [8]. Topic modeling is a frequently used text-mining tool for the discovery of hidden semantic structures in a text body. The topic will be extracted from each review and then a cluster of words will be generated from derived topics.

There are so many movies and tv shows coming out every week, that it can be difficult to keep up with it. It can take hours or even days to go through the blogs, posts, reviews to determine if a new piece of media is something that will be liked or not. With features established from topic modeling, a recommendation system for movies and TV series will be generated based on both content and user ratings. The problem is that the reviews are written in long textual formats and it is difficult for the users to read such long reviews and analyze topics from them. LDA and NMF will be used to extract topics from the reviews and then the results obtained will be visualized.[11,13]

Problem Statement: There are a lot of textual reviews and ratings available across different platforms for movies and TV shows. It is difficult to analyze thousands of reviews and recommend movies based on the reviews. In response to this problem, this project proposes building a recommender system that recommends movies and TV shows based on the ratings and topics discussed in the textual reviews.

Project Objectives: The dataset contains textual reviews for different types of movies and TV shows which allows performing analysis on it. Some of the objectives achieved in this project are:

1. An initial data exploration using Python to see and compare the trends in the dataset and gain insights from the information.
2. Extracting topics from the textual reviews using Latent Dirichlet Allocation(LDA) and visualizing them using pyLDAvis that helps users to interpret the topics discussed in the topic model that has been fit to a corpus of text data
3. Building a recommender system using extracted topics and ratings provided in the dataset.

The rest of the report focuses on literature review, initial data exploration, word cloud creation, and insights into the dataset, topic modeling using LDA, and building of recommendation systems. Finally, it also provides the conclusion and future scope for this project.

2.Literature review

Recommendation plays an important role in any web-based application for both customers and the product owners. It helps in increasing the efficiency of application and also in increasing customer satisfaction. There are various algorithms available for building recommender systems and the most popular are content-based recommender systems that are dependent on item features and Collaborative Filtering based recommender systems. [15]. The content-based recommender system gives recommendations depending on the item features. Collaborative Filtering based recommender systems provide recommendations that are similar to items in the user's history of selection. According to Hasanzadeh et al., about 88% of the users trust online written reviews. The paper published by Hasanzadeh et al proposes a recommendation system that recommends the movies based on ratings given by users. Initially, the preprocessing on the text is conducted which includes converting all the text into lower case and applying lemmatization and tokenization on the dataset. In the last step, based on the reviews recommendation system is developed where reviews act as a key feature and previous ratings acts as user's preferences.

Li Chen et al (2014) in their paper discussed the framework for building recommender systems from user reviews [12]. According to this research, four steps are involved in building recommender systems and they are: review analysis method, review element, user/product profile and recommender approach. It takes the example of a hotel review from a trip advisor and divides the review into feature opinions, comparative opinions and contextual information.

Bin Shai et al (2017) created a Stack overflow answer recommender with the help of LDA model[13]. The dataset available on stack exchange was used in this paper. Dataset was downloaded in XML format and then converted to MySQL. With the help of LDA, topics were extracted from the questions and those topics were used to cluster the similar questions together. The main approach was divided into 2 steps:Extracting features and predicting approach. Hence the features are extracted first and then any new question is fed to the LDA model that is previously trained. Then the new question gets a topic number which represents the cluster in which the question is divided.

Bin Chui et al (2014) created a location-content-aware recommender system called LCARS. This system was developed to answer 2 important questions based on user activity history: (1) What are the venues worth visiting in any city? (2) If we want to attend local events such as dramas, exhibitions in a city, where should we attend?[14]. Initially, the offline model was designed to model the user preferences to spatial items. Then the online recommendation is presented based on the top spatial items.

Younghoon Kim et al (2013) developed a system called TWILITE. It is a model for twitter built using probabilistic modeling based on LDA [15]. First, a model using probabilistic modeling based on LDA is built. Then, an inference algorithm for estimating the topic preference distributions of the users is established. Lastly, the recommendations were done using top-k recommendation algorithms where users were given the choice to select followers and tweet messages.

While building the recommendation system for answering stack overflow questions, Shao clustered all the questions based on the topics obtained from the topic modeling. In this project a similar approach is applied where topics are clustered and a theme is generated after the application of LDA. Similarly, as implemented by Chen et al. Collaborative Filtering is applied based on the title and ratings given by the user. Recommendations are made based on the previous interests and ratings provided by the users.

3. Initial Data Exploration and Visualization

This section is divided into various sections explaining the overview of the dataset. Initially, it provides information about the dataset, attributes of the data followed by basic visualizations using bar charts. The next section describes the generation of word clouds and the distribution of reviews. Then, data cleaning and preprocessing of the data are explained which will be helpful in the LDA model. Lastly, it includes a description of metadata and attributes selection from metadata. The link to the dataset is https://jmcauley.ucsd.edu/data/amazon/

3.1 Data Acquisition

The dataset used for this project is available on Julian McAuley's educational website along with the description of the dataset. It contains various Amazon review datasets for Electronics, Movies, and TV, Books, etc. Amazon Movie and TV reviews dataset was chosen for this project. The dataset contains around 142.8 million reviews. It is difficult to perform LDA on such a huge dataset due to lack of RAM so I decided to only take half of the dataset for this project.

The dataset is in the JSON file format zipped into a gz file. It contains the product reviews and metadata from May 1996-July 2014. The reviews and metadata for the reviews are present according to the categories. This is an unstructured dataset and is stored in JSON objects. It also additionally contains reviews(ratings, text, helpfulness votes), product metadata(descriptions,

category information, price, brand, and category features). Both data and metadata are used in this Capstone project for building recommendation systems.

3.2 Dataset Description

Table[1] contains a detailed description of the dataset used in this Capstone project along with data types.

Variable	Definition & Relevance
reviewer ID	Unique ID for a Reviewer - Relevant
ASIN	Amazon Standard Identification Number. The ID of the product - Relevant
reviewerName	Name of the Reviewer - Irrelevant
helpful	Helpfulness Rating of review - Relevant
reviewText	Textual Review - Relevant
overall	Number of stars a reviewer has chosen to rate the product- Relevant
summary	Summary of the review - Relevant
unixReviewTime	Time of the review(Unix Time)- Irrelevant
reviewTime	Time of the review submitted in standard mm-dd-yy format(raw Time)- Irrelevant

Table [1] shows the description and relevance of each attribute in the dataset.

3.3 NaN values treatment

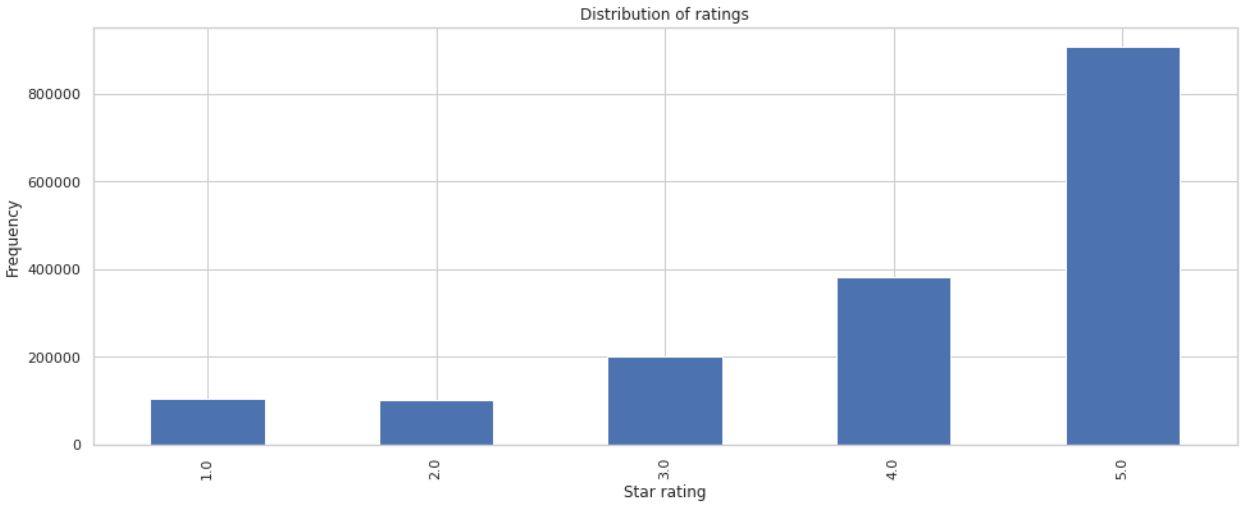
Some rows had NaN values for some of the attributes. Since the dataset is big enough and the NaN values were very few in comparison with the dataset, hence the rows with NaN values were

dropped. There was no missing data and hence no other cleanup was required before moving to Exploratory Data Analysis.

3.4 Data Exploration and Analysis

a) Distribution of Ratings

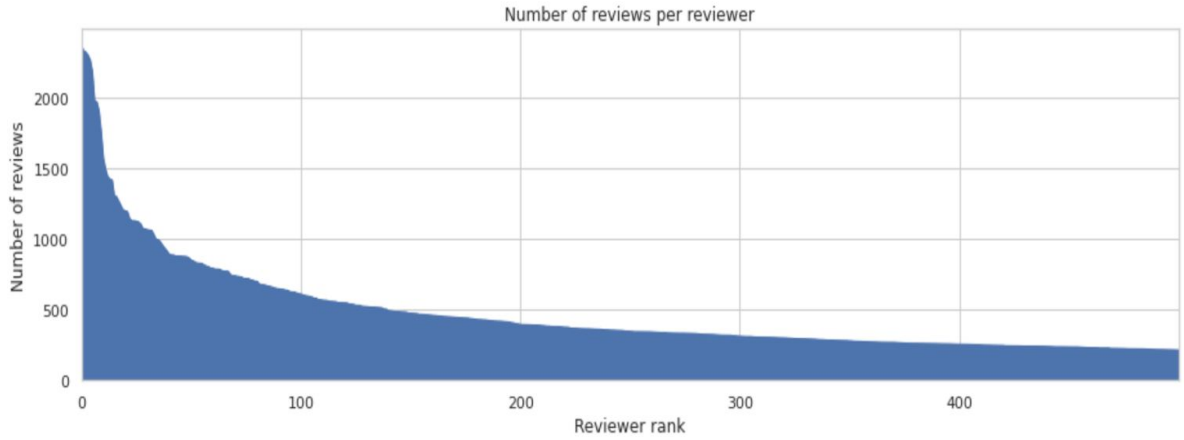
The reviews are rated between 1-star and 5-start with larger values corresponding to more favorable sentiment towards the movie. The distribution of ratings as seen in **Figure[1]** for reviews were plotted using a bar graph and as it can be seen in the graph that the ratings are heavily skewed toward 5-star ratings. Hence the average of all ratings is 4.11.



Figure[1] Distribution of Ratings

b) Number of reviews per reviewer

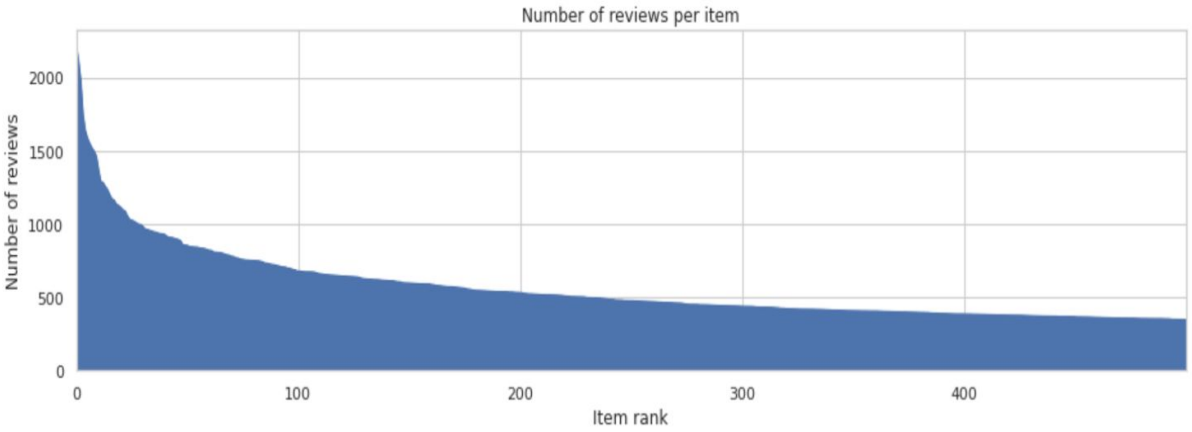
Next, in **Figure[2]** the number of reviews written by each reviewer is calculated and sorted based on each reviewer in descending order by their number of reviews written. It is observed that there is a long-tail distribution of reviews where few reviewers are highly large in number and the vast majority of reviewers write very few reviews. It can be concluded at this point that our dataset is processed correctly after observing this graph.



Figure[2] Number of reviews per reviewer

c) Number of reviews per item

To get an overview of the reviews per movie and TV shows, the number of reviews per item is plotted as shown in **Figure[3]** using a bar graph. First, reviews are calculated per item by grouping them on asin ID, and then they are sorted in descending order. Finally, the number of reviews per item is plotted against Item rank.



Figure[3] Number of reviews per item

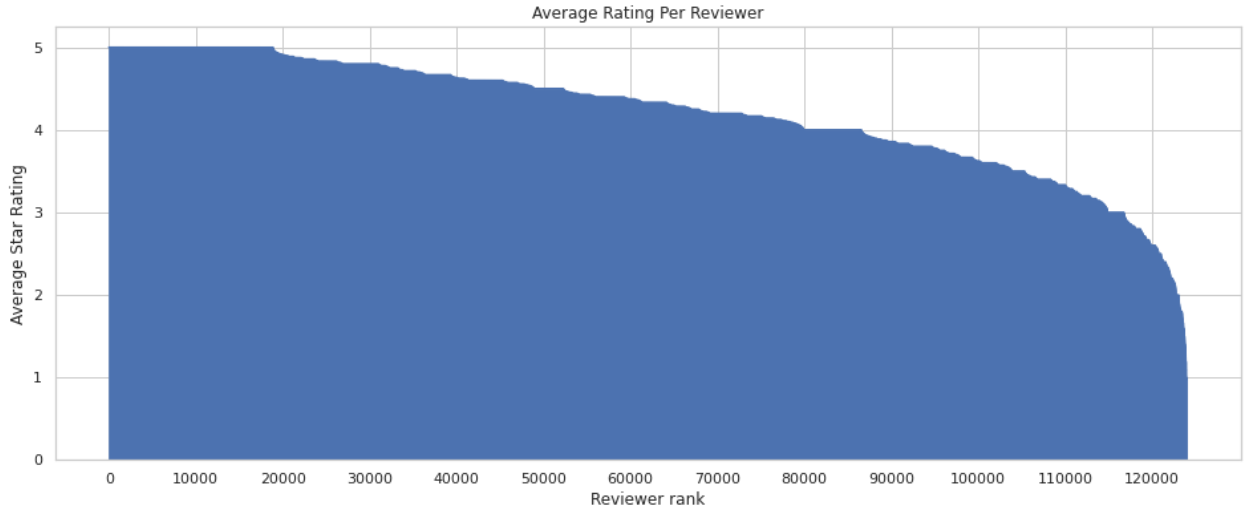
Again, it is observed that there is a long-tailed distribution of the number of reviewers against the number of reviews per item.

d) Average rating per reviewer

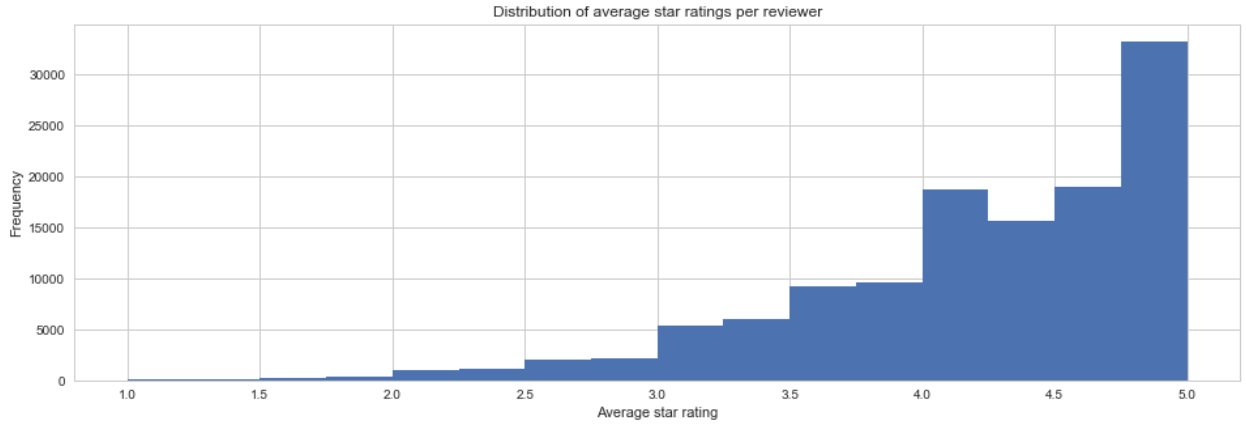
In this step, we calculate the average star rating per reviewer. First, each reviewer is sorted in descending order by their average star rating and then the area plot is plotted for the reviewer's

sorted rank against their average star rating. It is observed in the graph that about 90% of the ratings provided by reviewers are above 3-star.

After observing the area plot in **Figure [4]**, it makes sense that reviewers give 3-star ratings for the movies and TV shows that they watch because they are most likely to purchase the movies that they already like or they believe they will like. The reviewer rank depicts the rank of the reviewers based on the rating provided by them.nThe average is also calculated and plotted in **Figure[5]** and from the graph and the typical reviewer rates items with an average rating of 4.2.



Figure[4] Average Rating per reviewer



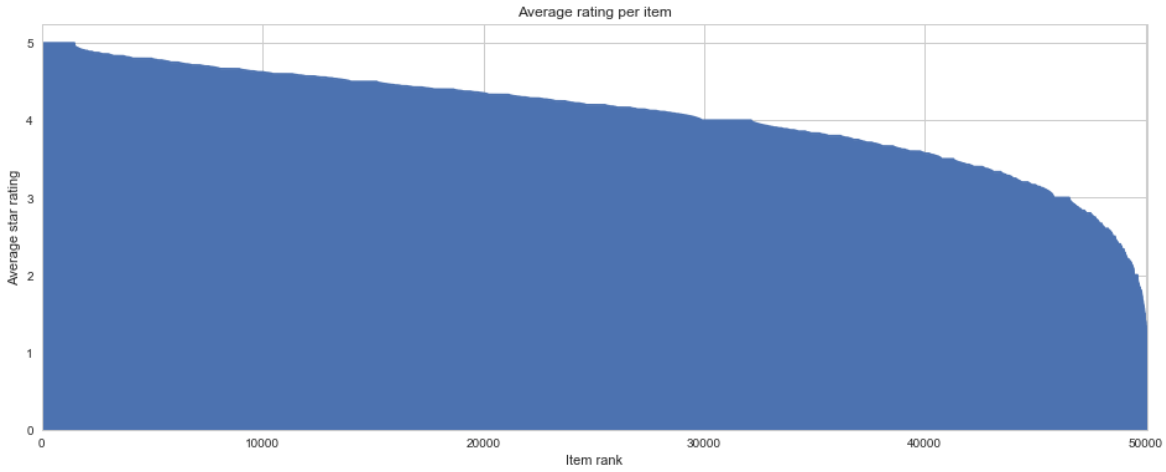
Figure[5] Distribution of average star rating per reviewer

e) Average rating per item

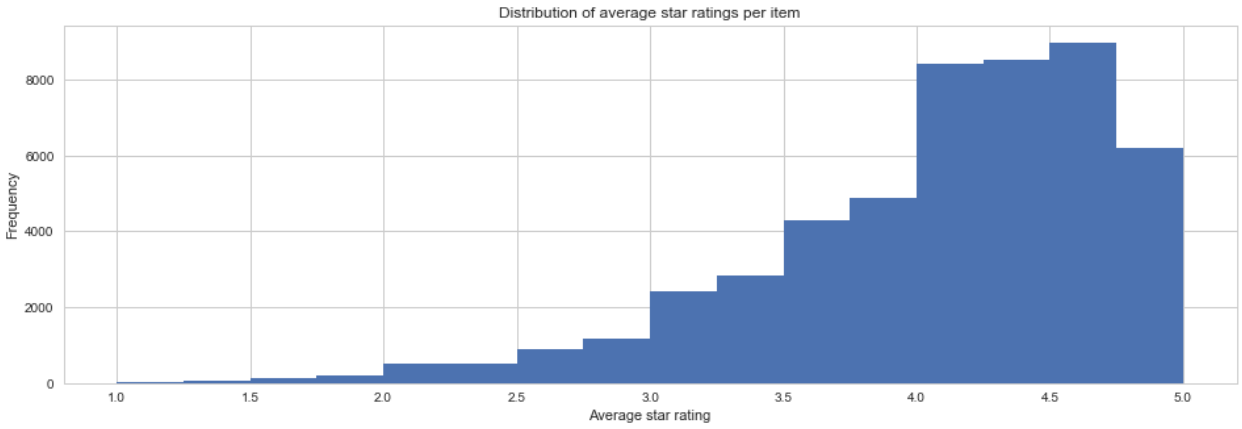
In **Figure[6]** area plot is plotted for average star ratings per item. It is observed in the area plot that the smaller proportion of movies have an average star rating of 5- stars and

a smaller proportion of movies have an average rating star above 3-star. Next, the histogram is plotted in Figure[7] for the distribution of average star rating per item.

The histogram in **Figure[7]** shows the difference in distribution more vividly. It can be seen that the mode of the distribution is between 4 and 4.5 stars.



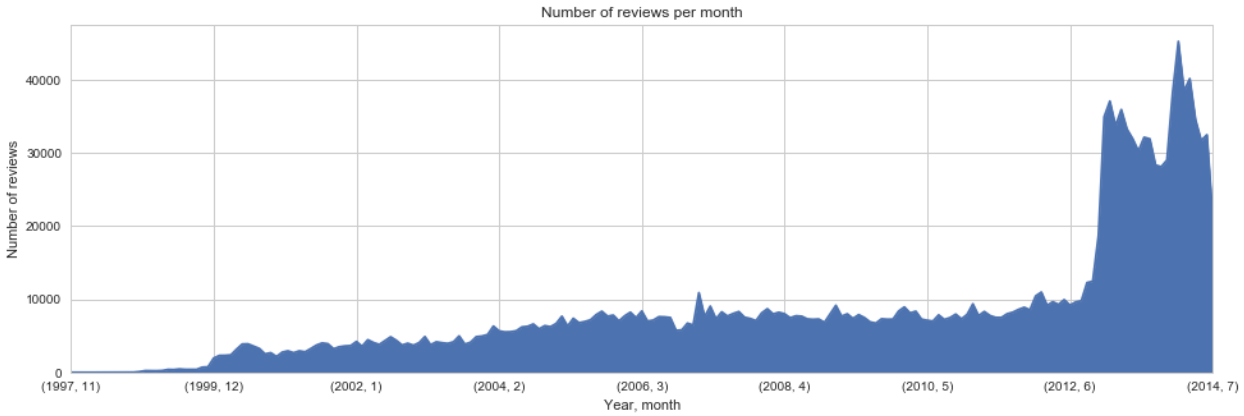
Figure[6] Average star rating per item



Figure[7] Distribution of average star ratings per item

f) Number of reviews over time

To get an idea of how many reviews are given by reviewers over a period of time(monthly), a graph is plotted in **Figure[8]**. It is clear from the graph that the number of reviews per month is consistent from December 1992 to June 2012. There is a considerable increase in the number of reviews after June 2012 because there was a deal between Amazon and Epix to add a large number of movies and TV shows on Amazon



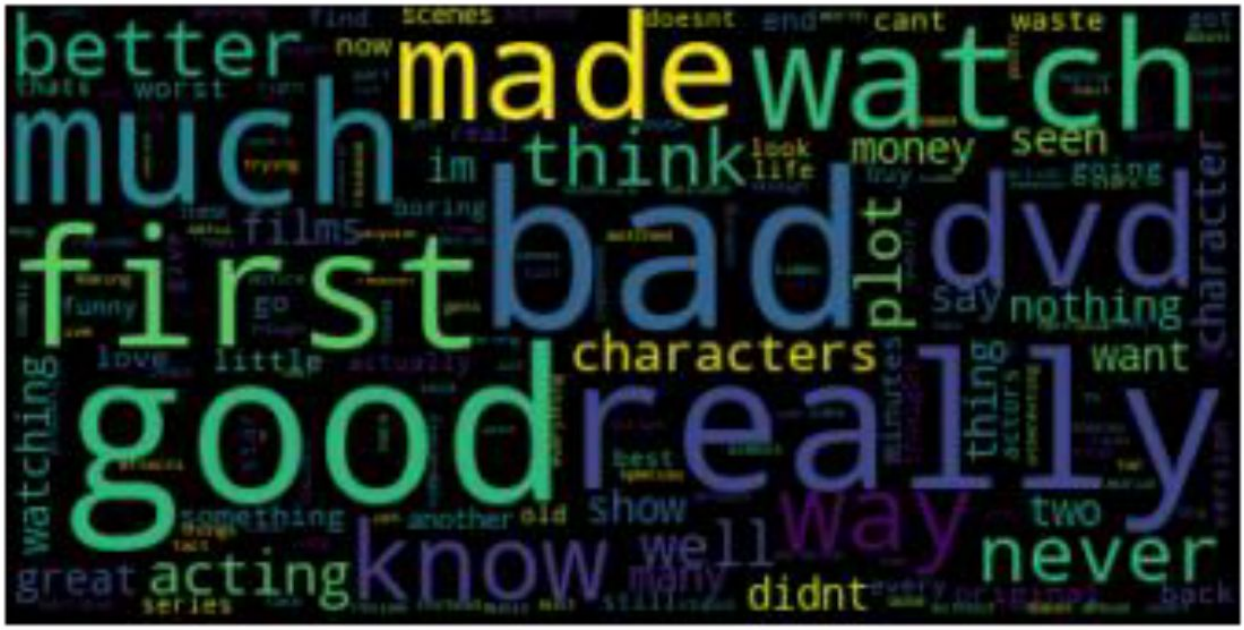
Figure[8]Number of reviews per month

4. Word Cloud

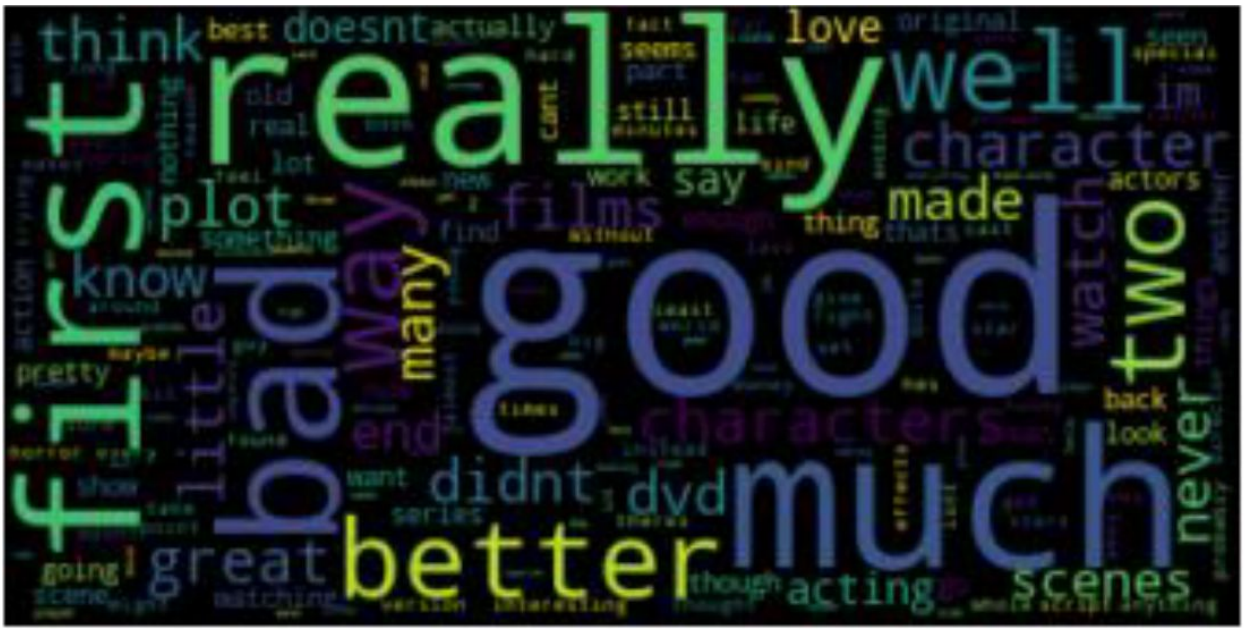
According to Google, Word cloud is an image composed of words used in a particular text or subject, in which the size of each word indicates the frequency and importance of that word in the overall dataset.[13] So, if the words appear bigger and bolder in the word cloud, the more often that word appears in the textual data.

The word cloud was generated based on the star ratings in the dataset and the most frequently appearing words can be observed in the word cloud. One of the biggest advantages of using word clouds is that it provides a visual representation of the frequency of the words appearing in the dataset. [18]The human brain loves the idea of visualizing text in appealing format and that is one of the most important reasons why word clouds are popular when it comes to representing the text.

In **Figure[9] , Figure[10], Figure[11], Figure[12] and Figure [13]** the word cloud for 1-stars , 2-stars, 3-stars,4-stars and 5-stars ratings is represented respectively. As it is clearly evident from the word cloud that we have processed data correctly. As observed in Figure[9], 'bad' words appear bigger as compared to other words because those movies are rated as 1-star. Also, in Figure[13] the word 'great' appears bigger as compared to other words in the word cloud. This reassures us that our data is processed correctly and the word cloud is generated appropriately.



Figure[9]Word Cloud for 1-star ratings



Figure[10]Word Cloud for 2-star ratings

Figure[11]Word Cloud for 3-star ratings

Figure[12]Word Cloud for 4-star ratings

Figure[13]Word Cloud for 5-star ratings

5. Latent Dirichlet Allocation(LDA)

LDA is a popular method in natural processing which is mostly used for topic modeling the textual reviews. [17,23]It is described as a process of extracting topics from the textual reviews. Topic modeling is the method used for unsupervised classification of textual documents. This process is similar to clustering on numeric data. LDA classifies the documents based on the similarity of the topics discussed in the texts.

Topic modeling is the method used for clustering similar words in the documents together. [22]This method is used for understanding, searching, automatically organizing large text in any format. The main aim of topic modeling is to:

- Identifying topics discussed in the documents
- Classification of documents based on the theme
- Using the above classification, summarize the documents

LDA is considered the most popular topic modeling methods.[25] There are various words in each document and each word belongs to various topics. Based on the words, LDA helps in finding the documents to which word belongs. There are a few assumptions that are necessary to make to implement LDA successfully.

5.1 Assumptions for LDA

Following are the assumptions that we need to make beforehand for successful implementation of LDA on the large text:

- The order of the words and grammar(verb, subject, etc) of the words in the document is not important because the document is just a collection of words also known as a "bag of words".

- Stopwords such as is, are, or, am, I, of, a, they are not relevant concerning the topic modeling and hence can be removed as a part of the preprocessing the data. These words do not add any value to the LDA.

- Depending on the length of data and text, we have to assume several topics beforehand. This can be fine-tuned as we built our model and depending on the output we get this number can be changed.

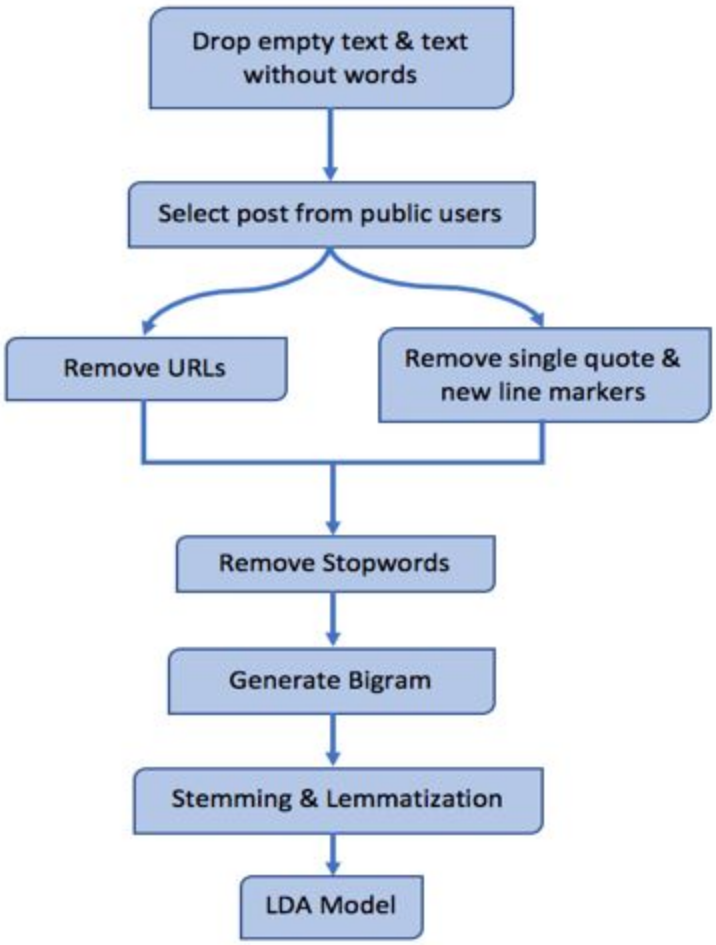
- It is assumed that all the topics assigned except for the current word are correct and based on our model, the assignment of the current word can be updated.

5.2 Data preprocessing for LDA

Some preprocessing on data is necessary before applying the LDA algorithm. In this project, the 'nltk' python library is used to preprocess the text. As seen in **Figure[14],** the data preprocessing for LDA involves various steps. This flowchart published by Shuaidong, et. al gives an overview of the steps that are performed on the textual data before the application of the LDA algorithm.

Step 1 involves **extracting the dataset** from the source which in this project is Julian McAuley's web page. The dataset is read into the data frame using python which makes performing operations on it easier.

Step 2 is to **convert to lowercase, tokenize, and remove stopwords** from the text. All the text is first converted into lower case to avoid any discrepancies related to case sensitiveness. Then all the punctuations such as comma, semicolon, exclamation mark, single quotes, double quotes are removed from the text since they do not play an important role in LDA. Lastly, a tokenizer is used to split all the sentences into words.



Figure[14]Flow chart for LDA preprocessing

Step 3 is to perform stemming and Lemmatization on the words. Stemming and lemmatization are used to generate the root form of the word. For example, “running” becomes “run” after stemming and Lemmatization is applied.

Step 4 involves incorporating all the preprocessed data from the above steps and creating a list of documents (also known as a list of lists). Then lastly a document term matrix is created which gives us the frequency of each term with each document.

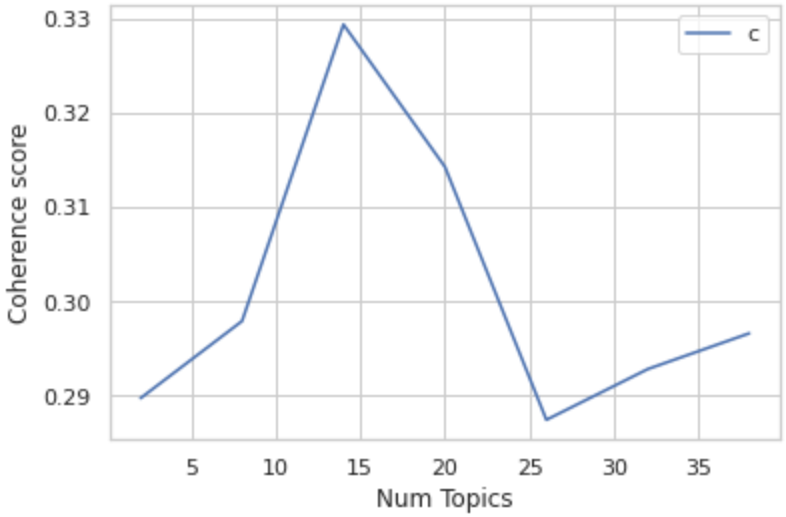
5.3 Implementation of LDA

Once all the data is preprocessed and ready, an LDA algorithm is applied to the clean dataset. The next step is to create a term dictionary for our generated corpus in the previous step. The term dictionary is used to assign an index to every unique term in our corpus. For assuming the

number of topics beforehand, coherence technique is used. Before applying LDA on the preprocessed data, it is important to determine the number of topics that would be assumed.

Topic Coherence measures score a single topic by measuring the degree of semantic similarity between the high scoring words in the topic. Such measurements are helpful in distinguishing between topics that are semantically interpretable topics and topics that are artifacts of statistical inference

If the set of statements or facts support each other they are known as coherent. To determine the appropriate number of topics to be assumed, initially coherence measure is calculated and coherence score is plotted against the number of topics. Figure[15] shows the graph of coherence score against the number of topics initially assumed.



Figure[15] Topic Coherence graph

After taking a closer look at the graph, it is observed that coherence score seems to keep increasing with a decline between 26 to 35. Hence it makes sense to choose the number of topics that yield maximum coherence score. In this case we choose 15 as the number of topics. In the last step, genism's LDA model is trained and implemented by passing the number of topics as 15.

Figure[16] shows the topics generated by LDA along with the frequency of the top 10 words associated with the topics.

```
Topic: 0
Words: 0.026*"time" + 0.026*"excellent" + 0.022*"acting" + 0.021*"story" + 0.021*"western" + 0.021*"watch" + 0.019*"actor" + 0.017*"class
ic" + 0.015*"favorite" + 0.014*"cast"

Topic: 1
Words: 0.030*"life" + 0.013*"man" + 0.010*"son" + 0.010*"live" + 0.008*"father" + 0.008*"wife" + 0.008*"come" + 0.008*"love" + 0.008*"wom
an" + 0.008*"leave"

Topic: 2
Words: 0.066*"book" + 0.033*"read" + 0.019*"story" + 0.017*"novel" + 0.014*"like" + 0.012*"version" + 0.012*"adaptation" + 0.011*"charact
er" + 0.009*"think" + 0.009*"watch"

Topic: 3
Words: 0.031*"horror" + 0.013*"scary" + 0.013*"like" + 0.011*"zombie" + 0.011*"vampire" + 0.009*"watch" + 0.009*"dead" + 0.009*"time" +
0.008*"halloween" + 0.008*"iron_man"

Topic: 4
Words: 0.058*"match" + 0.012*"scott" + 0.012*"like" + 0.012*"win" + 0.011*"time" + 0.009*"chocolat" + 0.007*"michael_caine" + 0.007*"acto
r" + 0.006*"football" + 0.006*"george"

Topic: 5
Words: 0.012*"action" + 0.007*"violence" + 0.006*"like" + 0.006*"way" + 0.005*"scene" + 0.005*"man" + 0.005*"criminal" + 0.004*"world" +
0.004*"time" + 0.004*"crime"

Topic: 6
Words: 0.033*"family" + 0.019*"daughter" + 0.015*"father" + 0.014*"mother" + 0.013*"life" + 0.008*"mary" + 0.008*"story" + 0.008*"love" +
0.007*"play" + 0.007*"ray"

Topic: 7
Words: 0.023*"western" + 0.022*"sam" + 0.017*"noir" + 0.009*"man" + 0.008*"gun" + 0.008*"jamie" + 0.008*"play" + 0.008*"star" + 0.007*"to
wn" + 0.007*"like"

Topic: 8
Words: 0.026*"story" + 0.020*"love" + 0.016*"character" + 0.015*"beautiful" + 0.015*"time" + 0.011*"feel" + 0.011*"life" + 0.011*"truly"
+ 0.009*"actor" + 0.009*"performance"

Topic: 9
Words: 0.024*"performance" + 0.019*"character" + 0.015*"role" + 0.012*"play" + 0.011*"actor" + 0.007*"work" + 0.007*"cast" + 0.006*"scen
e" + 0.006*"director" + 0.006*"time"
```

Figure[16] Topics extracted from LDA

The topics are generated along with the word frequencies. All of the themes identified by LDA are easy to understand. They are also easy to label such as topic 5 talks about “action”, “violence”, “criminal” can be clearly labelled as action/crime movies. In this way each topic can be interpreted and a theme is built which will be used to build a recommender system in the next part of the project.

6. Recommendation System

It can be seen on websites like Amazon, Netflix, Hulu, there is a section for “Shows you may like” This is nothing but the recommendations given by the websites which are produced using recommender algorithms.[14] As the name suggests, recommendations systems are useful in suggesting relevant items to the users. The recommendation systems are a popular concept amongst users. It helps them to choose the items or products that most likely align with their interests. The main purpose of the recommendation system is to suggest relevant items to users. There are 2 major algorithm categories for building recommendation systems and they are

Content-based algorithms and Collaborative Filtering methods.[24] In this project, the Collaborative Filtering Recommendation system is built using Matrix Factorization.

6. 1 Collaborative Filtering

Collaborative Filtering is the method popularly used to build recommendation systems. This method. It is based on the interactions recorded by users in the past. The new recommendations are produced based on the reviews or ratings given by users. All the ratings and reviews are stored in the "user-item interaction matrix".

As seen in Figure[17] the reviews are extracted from the source and are stored in the matrix. This matrix is used in building the recommendation system. Collaborative filtering assumes that the past ratings are sufficient for providing recommendations in present.

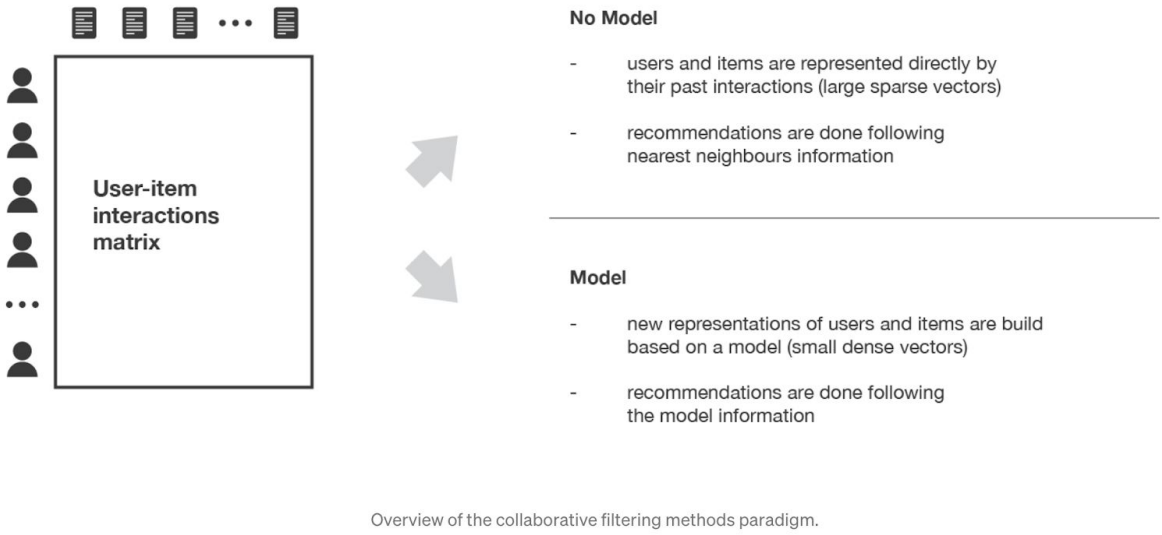
Users	User-item interactions matrix	Items
suscribers	rating given by a user to a movie (integer)	movies
readers	time spent by a reader on an article (float)	articles
buyers	product clicked or not when suggested (boolean)	products

Figure[20] Collaborative Filtering
Source: Adapted from [27]

This Collaborative Filtering is further divided into 2 sub approaches namely: Memory-based approach and Model-based approach. To make new recommendations, memory-based approaches do not follow any model. The recommendations are based on values recorded by users in the past. Some nearest neighbor search is applied in a memory-based approach and recommendations are provided to the users. The model-based approach follows some underlying model for providing recommendations. To make new recommendations, in a model-based

approach a model is built that produces some kind of relationship between the user-item interactions based on the dataset.

Figure[17] gives an overview of Collaborative Filtering methods that are followed while building recommendation systems. It is clearly evident from the figure that model-based approaches use small dense vectors and large sparse vectors are used in memory-based approaches. One of the biggest advantages of Collaborative Filtering is that it does not require any information about users and items. It only depends on the interaction between users and items. Hence, the more the user interacts with the item, the more accurate recommendations are generated.



Figure[21] Collaborative Filtering Methods
Source: Adapted from [28]

6.2 Architecture of Recommendation System

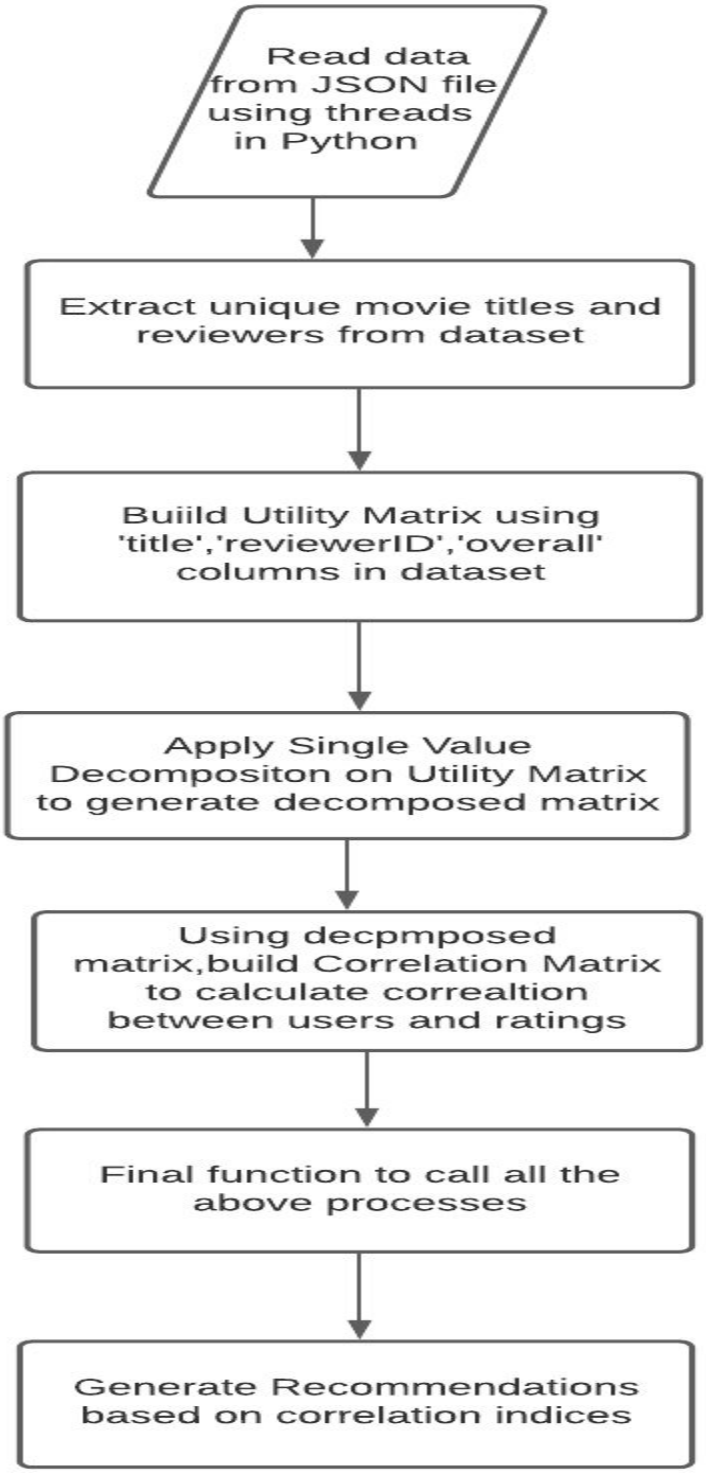
Collaborative Filtering:

Collaborative Filtering refers to the building a recommendation system using ratings and movie titles. There are various methods to implement collaborative filtering and in this project Matrix Factorization(SVD). Collaborative filtering is used to filter out the items that a user might like on the basis of reactions by similar users. The basic idea is to search a large group of dataset and find a smaller dataset with the same set of likings.

The steps used to build a recommendation system are discussed below:

In step 1, two threads are used to read data simultaneously as it makes reading a large amount of data faster and the data is stored in a dataframe. Only the columns required to build Collaborative Filtering models are extracted from the dataframe and all other columns are dropped.

In step 2, unique movie titles and unique reviewers in the dataset. The next important step is to build a utility matrix. Utility matrix is generated using the pivot table function in Python. The columns that are used to generate the utility matrix are ‘title’,’reviewerID’ and ‘overall’. The input is grouped into a two-dimensional table generating the multidimensional summarization of the data. Each row represents the user and corresponding column represents the item associated to that user



Figure[17] Architecture of Collaborative Filtering Recommendation System

In step 3. Single Value Decomposition(SVD) is applied using a python function on the utility matrix obtained from the previous step. SVD is the technique used to reduce the number of features of the data set without changing the dimensionality of the matrix. It is important to choose an appropriate number of latent factors while applying this technique. For this project, initially various numbers of latent factors were tried and output was observed to see if it makes any sense. Finally, 1000 latent factors gave good output and hence it was chosen to apply SVD. In layman's terms it is a process to find 2 matrices whose product equals the original matrix. The decomposed matrix is generated after SVD is applied to the utility matrix.

In step 4, a correlation matrix is generated using a decomposed matrix. Correlation matrix as the name suggests gives correlation between the users and the ratings and based on the past ratings. In the final step, a function is written to take the user ID as an input and based on the correlation matrix, it returns the movie recommendation based on the highest correlation indices.

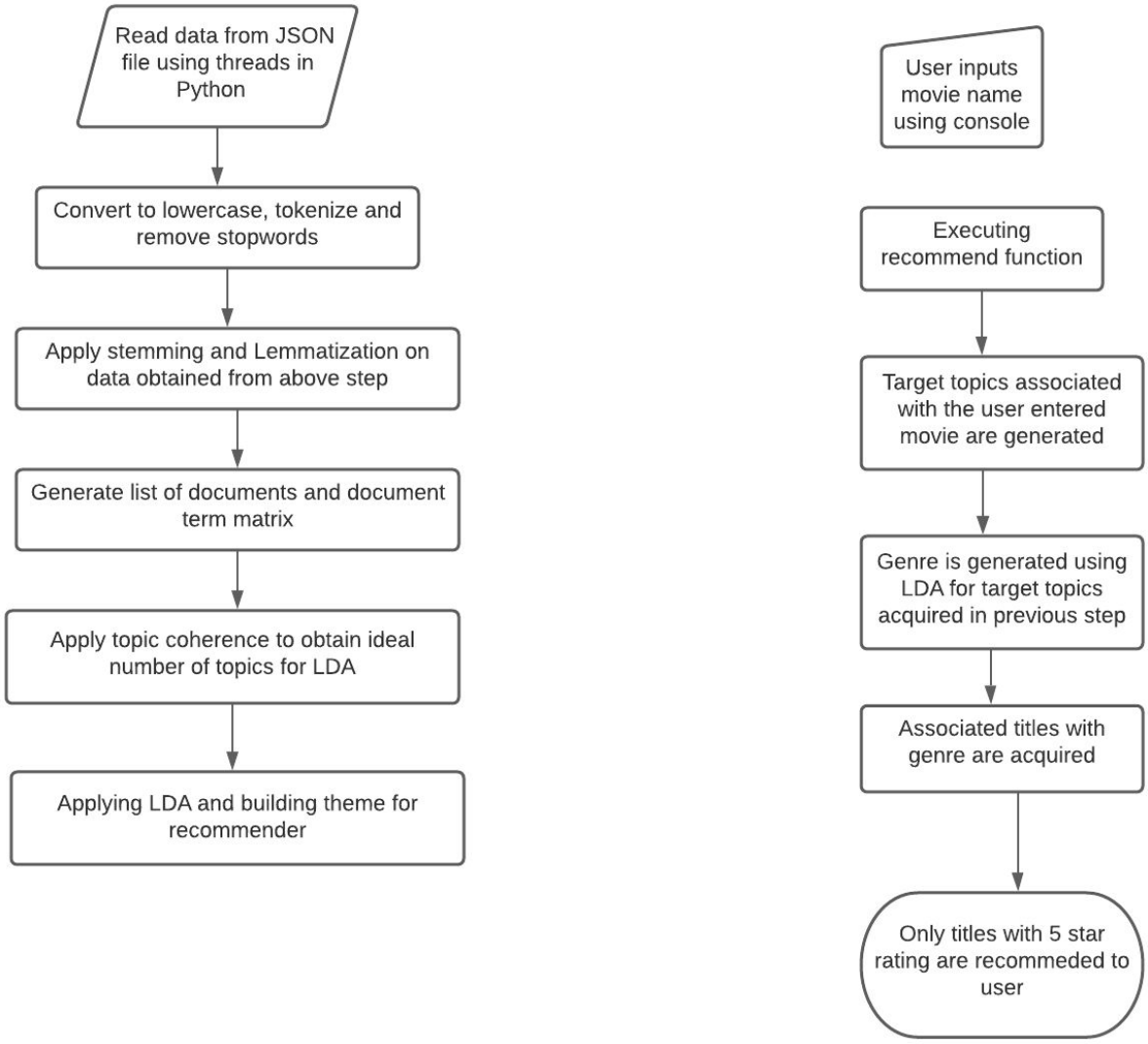
Theme based LDA recommendation:

In this type of approach, initially the reviews are tokenized and lemmatization is applied on them. All the stop words are removed from the textual reviews and finally when the dataset is ready, LDA is applied to generate the topics. The general approach as seen in Figure[17] is to perform text analysis and text mining on the dataset and make data ready for further analysis and development of the recommendation system. Text mining refers to the cleaning of dataset, dealing with missing values and applying techniques like lemmatization, tokenization and removal of stopwords in case of theme based LDA recommendation system. The clean review obtained after application of the above steps is known as 'bag of words'.

After the dataset is cleaned and text mining is completed, a review element is decided which in this case is textual reviews. Once the LDA is implemented and topics are generated, the quality of topics is determined and if needed some kind of fine tuning of topics is performed by determining the coherence score and changing the number of topics. Once the topics are generated they are considered as a review element and in this case a theme(genre) is built based on the topics. The reason why the theme is generated is explained in the later part of the report.

Once the topics are good, they are used to build a recommendation system. Due to scope of this project, there is no GUI for user input but it can be built in future. Following are the steps applied to build the recommendation system using theme based LDA:

In **Step 1** the user is asked to enter the name of the movie that they have already watched. This named is associated with the ID in the dataset



Figure[19] Architecture of Theme based LDA Recommendation System

In **Step 2** the recommendation function is called with the name of the movie as the parameter. This function then calls all other functions in the project.

In **Step 3** once the function has the name of the movie, it will try to look for the review associated with the movie and after the review is found, the target topics are generated. Basically, the target theme is acquired from the output of LDA.

In **Step 4**, all movie titles associated with the target topics are generated and stored in the list despite the ratings associated with them. There is a large number of titles that are generated after this step and hence it is not a good idea to display all of them to the user

In the last step, only the titles that have a 5 star rating are displayed to the user. This allows our model to only display the titles that have been highly rated and are highly liked by other users. Due to scope of the project,there is no GUI for the model. Users are asked to enter the ID or title directly via the console in Python and then the program runs and displays the output.

6.3 Implementation of Collaborative Filtering

Collaborative filtering using Matrix Factorization is implemented in this project. For this method, only overall ratings and movie titles are considered for recommendation. Initially, a matrix is built where rows represent each unique movie and TV shows titles and columns represent each unique user. [20]In the next step, the Singular Value Decomposition(SVD) is applied to reduce the dimensionality of the latent factors.

Singular Value Decomposition(SVD) states that any matrix can be represented as a product of 3 matrices that helps in reducing the dimensionality of the original matrix. [19,20]Initially, unique movie titles and reviewers are calculated from the dataset. Then we initialize the SVD class to decompose the utility matrix for 1000 most important latent factors.

In the next step, the correlation matrix is calculated from our decomposed matrix. In the last step, a function is created that will take user ID as an input and will return the top 5 movies based on the 5-stars that are calculated in the correlation matrix.In the final step, and web based application was developed using Flask and a simple GUI is displayed to the user. This application is deployed to Heroku.

6.4 Implementation of LDA based Recommender System

Based on the themes obtained from the LDA, recommendations using this system are made with the help of topics and themes extracted from the topic modeling. Initially, a bigram phraser from gensim is used to capture the predominant phrases from the documents.[16,19] Since the dataset is huge this helps in capturing the phrases that are useful for building recommendations. Once the phrases are obtained they are cleaned too by removing the stopwords and similarly applying Lemmatization techniques as data were preprocessed before applying the Collaborative Filtering Algorithm.

In the next step, a bag of words is created by parsing the file that contains documents obtained from LDA. Once the bag of words is obtained, it is passed to the recommend function in the code. [20,21]The recommendation function recommends the titles based on favorable reviews given by the user. This model does not allow fine-tuning of certain factors in the code and hence the output is not as efficient as expected.

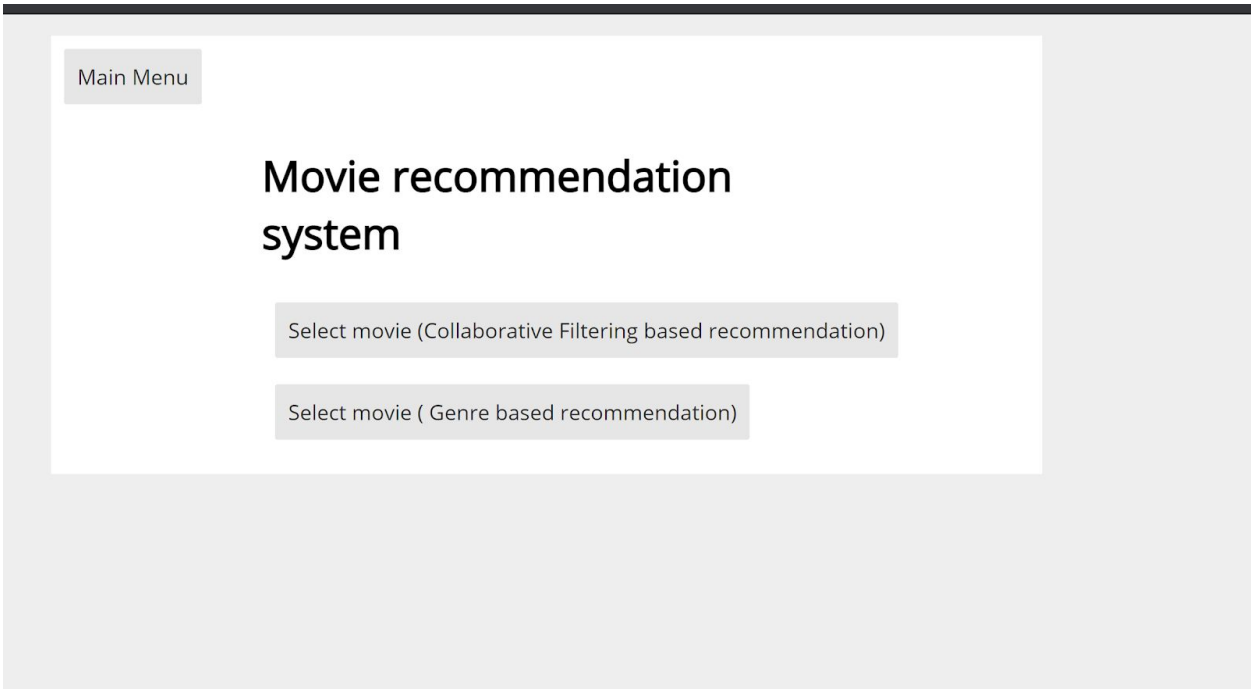
The whole idea of building a recommendation system using LDA is to generate a theme and then recommend movies based on the themes and favorable reviews given by a user.Hence when a title is entered by a user in this recommendation system the recommend_titles function is called. This function generates the target topics based on the title entered by the user from the theme based LDA. In the next step, the total titles matching the target topics are displayed.

After the target topics and titles are acquired, only a subset of titles (having rating =5) are displayed as recommendations to the user.

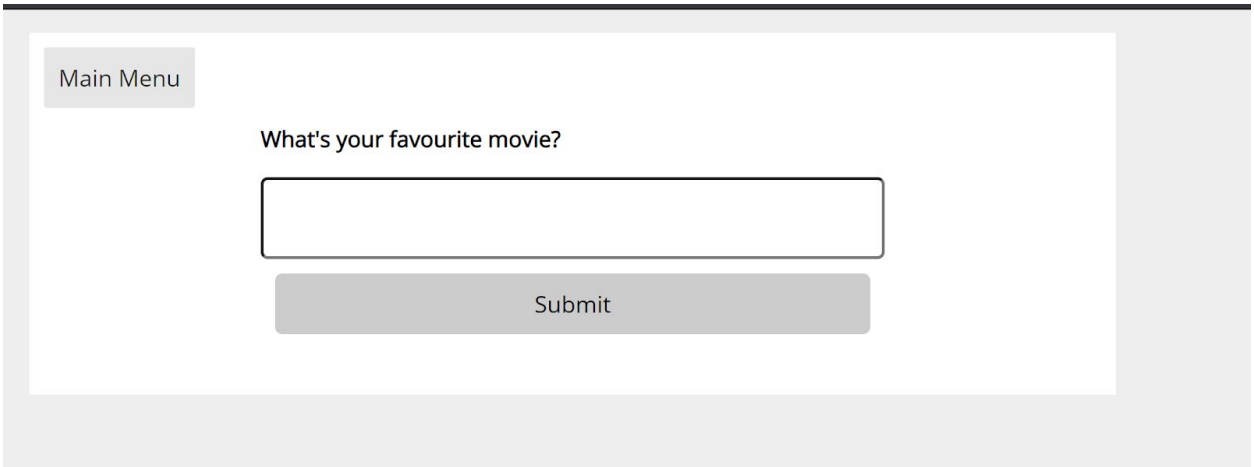
7. Results

The web application is available at https://amazonmovierecommendation.herokuapp.com.
This section includes the screenshots of the results. It shows how the GUI is displayed to the user and shows the results of both the types of recommendations developed in this project. Figure [20] shows the main menu that is displayed to the user. It gives the option to select what kind of recommendation user would like to try.

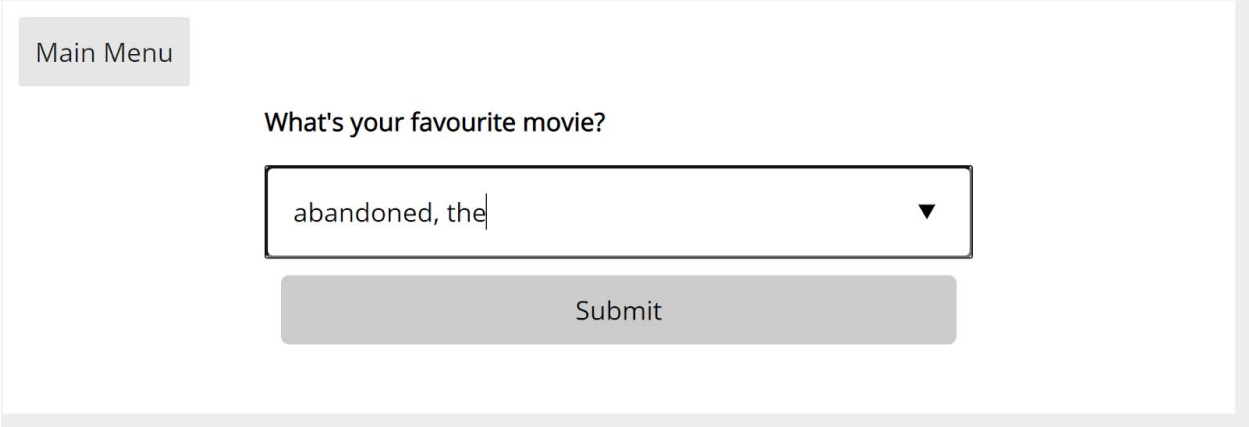
Once the option is selected, it will ask the user to select the title from the list which is displayed in Figure[21]. It allows the user to select the title and based on the selected title the recommendations are displayed to the user. Along with the recommendations, cosine similarity as well as genre similarity is also displayed to the user to give them idea about which titles are better as compared to others



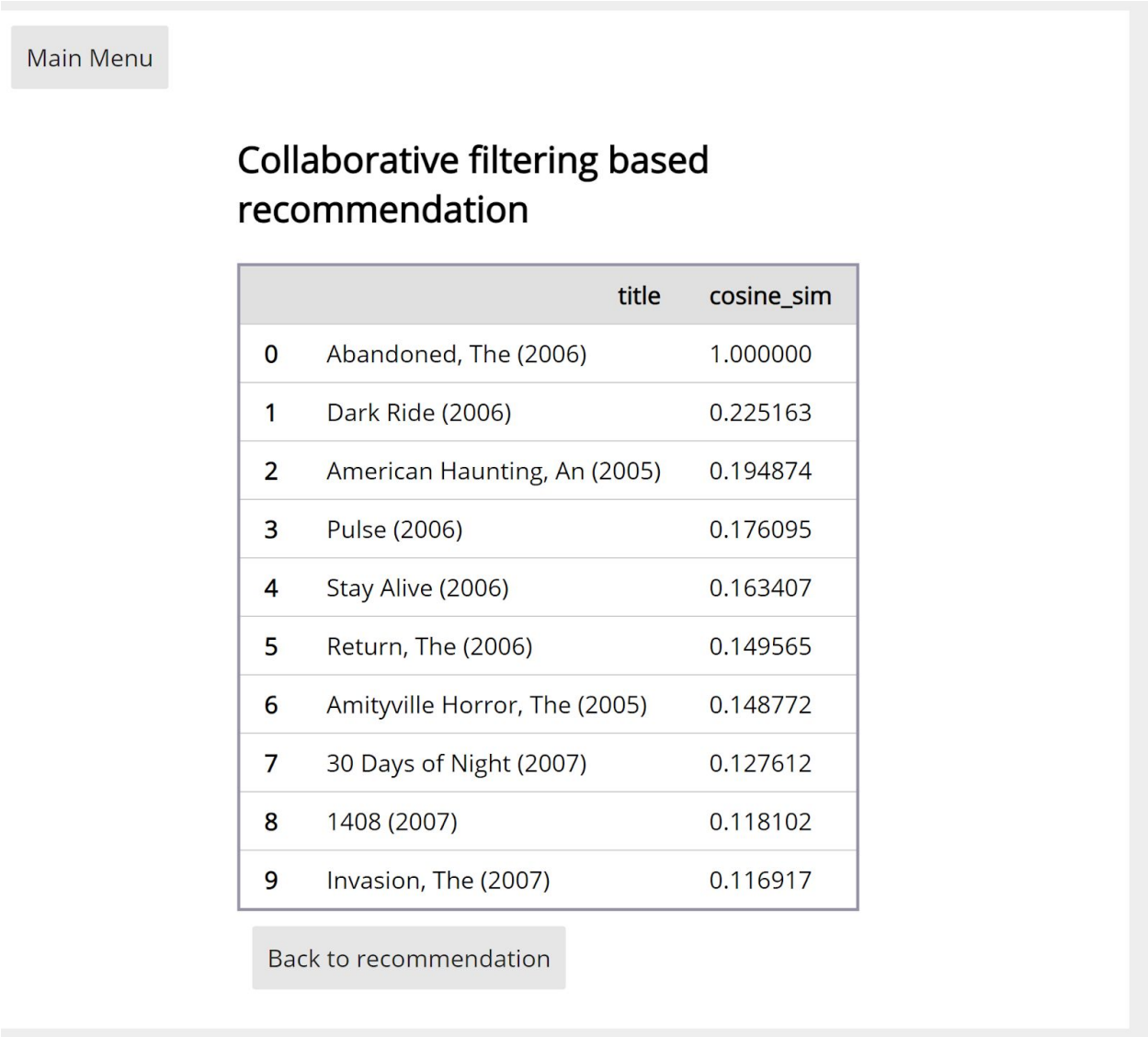
Figure[20] Main Menu displayed to the user



Figure[21] Enter the movie title



Figure[22] Enter the movie title



	title	cosine_sim
0	Abandoned, The (2006)	1.000000
1	Dark Ride (2006)	0.225163
2	American Haunting, An (2005)	0.194874
3	Pulse (2006)	0.176095
4	Stay Alive (2006)	0.163407
5	Return, The (2006)	0.149565
6	Amityville Horror, The (2005)	0.148772
7	30 Days of Night (2007)	0.127612
8	1408 (2007)	0.118102
9	Invasion, The (2007)	0.116917

Figure[22] Result of Collaborative Filtering Recommendation

Main Menu

Genre based recommendation

	title	cosine_sim	genre_similarity
0	Alpha Dog (2007)	1.000000	1.00000
1	16 Blocks (2006)	0.210098	0.50000
2	Apocalypto (2006)	0.193999	0.40825
3	28 Weeks Later (2007)	0.186112	0.00000
4	1408 (2007)	0.181792	0.40825
5	Déjà Vu (Deja Vu) (2006)	0.176673	0.00000
6	40-Year-Old Virgin, The (2005)	0.168040	0.00000
7	300 (2007)	0.167825	0.00000
8	30 Days of Night (2007)	0.166107	0.00000
9	I Am Legend (2007)	0.163854	0.00000

Back to recommendation

Figure[22] Result of LDA based Recommendation

8. Evaluation Metrics for Recommendation Systems

It is important to evaluate the recommendation systems built in the project to determine which model is best suited for the dataset used in this project. For evaluating recommendation systems

various metrics are available such as Mean Absolute Error(MAE), Root Mean Squared Error(RMSE), Precision, Catalog Coverage, and Runtime. For this project, only MAE is calculated to determine the effectiveness of recommendation systems. The second metric that is used to evaluate the recommendation system is to ask actual users to use and record their responses.

Mean Absolute Error:

MAE for Collaborative filtering model is calculated using the following formula:

$$\mathbf{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

Where Y_i is the prediction made by the recommendation system, X_i is the actual value expected, and n refers to the total number of observations in the dataset. The ratings column is used to calculate MAE for this model. For the scope of this project, a surprise library is used to calculate MAE. The custom dataset and the ratings for the titles provided by the recommendation system is given as an input for the library and based on this information MAE is generated.The MAE for Content Based Filtering(SVD) is generated as 0.6831.

User responses:

Actual humans were asked to try out the recommendation system built using both models. Users were asked to choose from the movies in the dataset and the recommendations were displayed based on the type of the system chosen. After that users were also asked to give ratings for the titles recommended

User Number	**Title chosen by user**	**Titles recommended By Collaborative Filtering**

User_1	Alice in Wonderland	Dumbo(3), Bambi(4), Mary Poppins(5), Lady and the Tramp(4), Snow White and the Seven Dwarfs(5)
User_2	Autumn in New York	Sweet November(3), Bounce(4), America's Sweetheart(5), Where the heart is(4), The Bachelor(5)
User_3	The Avengers	Lost in Space(4), Godzilla(5), Spawn(4), Sphere(4), The Jurassic Park:Lost World(5)
User_4	Aladdin	Beauty and the Beast(3), Lion King(4), Batman(5), Jurassic Park(5), Apollo 13(3)
User_5	Bad Boys 2	Bad Boys(5), The Fast and Furious(5), Underworld(4), Men in Black 2(5)

Table [2] User responses for Collaborative Filtering Recommendation System

Table [3] shows the user responses for LDA based Recommendation System. The Cosine Similarity and Genre similarity is also displayed to the user in order to give the better idea about which recommendations are good recommendations.

User Number	**Title chosen by user**	**Titles recommended**

		By LDA based recommender
User_1	Anastasia	Alice in Wonderland(5), All Dogs go to heaven(5), Pocahontas(3), Dumbo(3), The Little mermaid(4)
User_2	Batman Forever	Batman(5), True Lies(5), Die Hard(5), Stargate(3), Cliff Hanger(5)
User_3	Clueless	Pretty Woman(4), Sleepless in Seattle(5), Four Weddings and Funeral(5), Speed(4), While You were Sleeping(3)
User_4	Death Race	Jumper(4), Transformers(5), Resident Evil(5), X-Men(5), The Hulk(5)
User_5	Bad Boys 2	Bad Boys(5), The Fast and Furious(5), Underworld(4), Men in Black 2(5)

Table [3] User responses for LDA Based Recommendation System

8.Challenges

One of the most interesting challenges that were faced during the development of the recommendation system was the size of the dataset. Initially, it was not anticipated that this would be an issue in the processing of the dataset but when topic modeling in process, there was not enough RAM available in the Google Colab to process the dataset. The system crashed a couple of times because of the unavailability of the RAM and hence the dataset was reduced in half to successfully implement the LDA and build a recommendation system eventually. The second challenge was to understand and process the topics extracted from the dataset as a huge number of topics were generated which made it difficult to understand them.

9.Limitations

There are a few limitations to this project. First of all, these methods are only limited to this particular dataset because based on the dataset, a list of stopwords needs to be updated. Also, the topics that are generated from the topic modeling would change if the dataset is changed, the fine-tuning of LDA and assumption for the number of topics would change. There are a lot of assumptions made throughout the development of recommendations systems that may or may not hamper the outcome of the model. Another limitation is that since the dataset was cut into half due to lack of RAM, it is possible that some of the recommendations made by the system can be limited.

10.Conclusions and Future Scope

In this project, data is preprocessed and topic modeling is applied to the dataset. Initially, data is preprocessed and word cloud is generated. Once the topics are extracted from the dataset using LDA and these topics are stored in a text file. After extracting the topics, the recommendation system is developed using the Collaborative Filtering algorithm and using the topics generated from LDA. After comparing the evaluation metrics for both the algorithms, it is observed that Collaborative Filtering is better as compared to LDA based recommendation systems. In the future, a web-based application can be generated that will take movie titles from the user and will display recommendations along with the genre of the movies.

References:

[1] B. Shao and J. Yan, “Recommending answerers for stack overflow with LDA model,” in Proceedings of the 12th Chinese Conference on Computer Supported Cooperative Work and Social Computing - ChineseCSCW ’17, 2017, pp. 80–86.

[2] H. Yin, B. Cui, Y. Sun, Z. Hu, and L. Chen, “LCARS: A spatial item recommender system,” ACM Trans. Inf. Syst., vol. 32, no. 3, pp. 1–37, 2014.

[3] Y. Kim and K. Shim, “TWILITE: A recommendation system for Twitter using a probabilistic model based on latent Dirichlet allocation,” Inf. Syst., vol. 42, pp. 59–77, 2014.

[4] “Medium,” Medium.com. [Online]. Available: https://medium.com/@manjunathhiremath.mh/identifying-bigrams-trigrams-and-four-grams-using-word2vec-dea346130eb. [Accessed: 06-Nov-2020].

[5] Researchgate.net. [Online]. Available: https://www.researchgate.net/publication/337951950_Help_Oneself_in_Helping_the_Others_the_Ecology_of_Online_Support_Groups. [Accessed: 06-Nov-2020].

[6] Y. Heng, Z. Gao, Y. Jiang, and X. Chen, “Exploring hidden factors behind online food shopping from Amazon reviews: A topic mining approach,” J. Retail. Consum. Serv., vol. 42, pp. 161–168, 2018.

[7] L. Yu, C. Zhang, Y. Shao, and B. Cui, “LDA*: A robust and large-scale topic modeling system,” Vldb.org. [Online]. Available: http://www.vldb.org/pvldb/vol10/p1406-yu.pdf. [Accessed: 06-Nov-2020].

[8] C. Rana, “Building a Book Recommender system using time based content filtering,” Wseas.org. [Online]. Available: http://www.wseas.org/multimedia/journals/computers/2012/54-571.pdf. [Accessed: 06-Nov-2020].

[9] F. O. Isinkaye, Y. O. Folajimi, and B. A. Ojokoh, “Recommendation systems: Principles, methods and evaluation,” Egypt. Inform. J., vol. 16, no. 3, pp. 261–273, 2015.

[10] M. Z. Asghar, A. Khan, S. Ahmad, and F. M. Kundi, “A Review of Feature Extraction in Sentiment Analysis,” Statmt.org. [Online]. Available: http://www.statmt.org/OSMOSES/FeatureEx.pdf. [Accessed: 06-Nov-2020].

[11] Semanticscholar.org. [Online]. Available: https://www.semanticscholar.org/paper/Opinion-. [Accessed: 06-Nov-2020].

[12] “Medium,” Medium.com. [Online]. Available: https://medium.com/@m_n_malaeb/recall-and-precision-at-k-for-recommender-systems-618483226c54. [Accessed: 06-Nov-2020].

[13] L. Chen, G. Chen, and F. Wang, “Recommender systems based on user reviews: the state of the art,” User Model. User-adapt Interact., vol. 25, no. 2, pp. 99–154, 2015.

[14] H. Jelodar, Y. Wang, M. Rabbani, and S. Ayobi, “Natural language processing via LDA topic model in recommendation systems,” arXiv [cs.IR], 2019.

[15] Psu.edu. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.664.8706&rep=rep1&type=pdf. [Accessed: 07-Nov-2020].

[16] Y. Cao, W. Li, and D. Zheng, “A hybrid recommendation approach using LDA and probabilistic matrix factorization,” Cluster Comput., vol. 22, no. S4, pp. 8811–8821, 2019.

[17] S.-M. Choi, S.-K. Ko, and Y.-S. Han, “A movie recommendation algorithm based on genre correlations,” Expert Syst. Appl., vol. 39, no. 9, pp. 8079–8085, 2012.

[18] G. Lekakos and P. Caravelas, “A hybrid approach for movie recommendation,” Multimed. Tools Appl., vol. 36, no. 1–2, pp. 55–70, 2008.

[19] S. Bergamaschi and L. Po, "Comparing LDA and LSA topic models for content-based movie recommendation systems," in Lecture Notes in Business Information Processing, Cham: Springer International Publishing, 2015, pp. 247–263.

[20] J. H. Errico, M. I. Sezan, G. R. Borden, G. A. Feather, and M. G. Grover, "Collaborative recommendation system," vol. 8949899. pp. 03– 2015.

[21] S. Kinoshita, T. Ogawa, and M. Haseyama, "LDA-based music recommendation with CF-based similar user selection," in 2015 IEEE 4th Global Conference on Consumer Electronics (GCCE), 2015.

[22] C. C. Aggarwal, Data mining: the textbook. Cham Switzerland; New York: Springer, 2016.

[23] J. W. Uys, N. D. du Preez, and E. W. Uys, "Leveraging unstructured information using topic modelling," PICMET '08 - 2008 Portland International Conference on Management of Engineering & Technology.

[24] B. Ma, D. Zhang, Z. Yan, and T. Kim, "An lda and synonym lexicon based approach to product feature extraction from online consumer product reviews," J. Electron. Commer. Res., vol. 14, no. 4, p. 304, 2013.

[25] N. Vaidya and A. R. Khachane, "Recommender systems-the need of the ecommerce ERA," in 2017 International Conference on Computing Methodologies and Communication (ICCMC), 2017.

[26] G. Shani and A. Gunawardana, "Evaluating Recommendation Systems," in Recommender Systems Handbook, Boston, MA: Springer US, 2011, pp. 257–297.

[27] B. Pathak, *Illustration of the user-item interactions matrix.* [Online]. Available: https://miro.medium.com/max/1750/0*BVPWcf2WyqJ5a1g7.png

[28]B. Pathak, *Overview of the collaborative filtering methods paradigm.* [Online]. Available: https://miro.medium.com/max/1750/0*oYvODcDGcScCj4dT.png