Sentiment-Analysis-Based Movie Recommender Systems

DSC 672 Capstone Project

Group 3

Xiaojing Shen
Data Science – Computational Methods
Lake Orion, MI 48360
Email: xshen16@depaul.edu

Serena Yang
Data Science – Computational Methods
Chicago, IL 60610
Email: myang43@depaul.edu

Yue Hou
Data Science – Computational Methods
Chicago, IL 60527
Email: yhou14@depaul.edu

Word Count: 4050 words

Submitted [Mar - 17 - 2022]

ABSTRACT

IMDb (an abbreviation of Internet Movie Database) is an online database of information related to films, television series, home videos, video games, and streaming content, ratings and critical reviews. In the current era of big data, data scientists can use the information provided by IMDb to create more accurate movie tracking and recommendations for users. We propose to build a sentiment-analysis-based movie recommender system using data from IMDb. Most recommender systems apply static information such as movie types, directors, writers, and cast. Our analysis integrated not only static information but with reviews and ratings, which could contribute to highly personalized recommendations.

There are three milestones in this project. Sentiment analysis was first implemented on an IMDb dataset from Kaggle, on which multiple models will be applied to select the ideal arithmetic model. Next, we applied a web crawler on IMDb to crawl data that fit in the following recommender systems. The last step was integrating crawled data with the selected sentiment model to build a personalized recommender system.

Keywords: Sentiment Analysis, Web Crawler, Recommender System

INTRODUCTION

IMDb (an abbreviation of Internet Movie Database) is the world's most popular and authoritative source for movie, TV, and celebrity content. The database of IMDb contains more than 8.7 million titles, 11.4 million personal records, and 83 million registered users.

We built a sentiment-analysis-based recommender system based on data and information from IMDb. Sentiment analysis is contextual mining of text which could identify and extract subjective information in the source material. Recommender systems refer to systems that can predict the future preference of a set of items for users and recommend the top items. Most recommender systems apply static information such as movie types, directors, writers, and cast. Our analysis integrated both static information and reviews, which could contribute to highly personalized recommendations.

To build a sentiment-analysis-based recommender system, we divided jobs into three milestones. In the first milestone, sentiment analysis was implemented on an IMDb dataset from Kaggle, on which multiple models were applied to select the ideal arithmetic model. The models include *DNN, CNN-LSTM, Term Frequency–Inverse Document Frequency (TFIDF),* and *Bag of Words (BOW)*. Next, we applied a web crawler on IMDb to crawl data that fit in the following recommender systems. Considering vast numbers of reviews, we crawled only the top 250 movies from IMDb. The last step was integrating crawled data with the selected sentiment model to build personalized recommender systems. Collaborative filtering-based recommender and content-based recommender were first implemented. Then we applied a new method called rating-based recommender.

LITERTURE REVIEW

A paper on Netflix Recommender System (Gomez-Uribe et al., 2015) was reviewed in this project. There are eight kinds of recommendation systems in Netflix - Personalized Video Ranker, Top-N Video Ranker, Continue Watching, Trending Now, Video-Video Similarity, Page Generation: Row Selection and Ranking, Evidence and Search. From this article, we realized that people face an increasing number of choices in every aspect of their lives, including watching films. Therefore, the recommender system will play a pivotal role in helping them find an optimal movie. In addition, a recommendation system built by multiple algorithms will provide more options and avoid customers being trapped by what they believe. In our project, Sabarman's method will be referred to extract movie reviews from IMDb (Sabarman, 2020). Then, a movie recommendation system will be built by movie reviews without reading the reviews.

Rehman et al. suggests that CNN helps to learn how to extract features from the data (Rehman et al., 2019). Moreover, the LSTM model can capture long-term dependencies between word sequences. Those two techniques will bring benefits to natural language processing. The experiment results explain that hybrid CNN-LSTM models perform better than single neural network techniques. In addition, the accuracy of the hybrid CNN-LSTM model is higher than traditional machine learning techniques. Furthermore, it consumes less memory and produces better results in terms of accuracy. Therefore, we decided to apply the hybrid CNN-LSTM model in our project to compare with DNN or other models through ROC curve and accuracy.

An article from towardsdatascience.com introduced TFIDF from scratch (William, 2019). Starting from the formula of this method, term frequency, and inverse document frequency were

first introduced in detail. Following the introduction, real-world data was implemented using TFIDF. All the details, including data preprocessing and calculation of TFIDF, were included; therefore, this paper is an excellent example to follow when we implemented a similar method.

For stage 2, to get a dataset with all the information we need for stage 3, we learned how to crawl the IMDB website using BeautifulSoup from Breuss's article (Breuss, 2021). The article mainly covered the inspect data sources, scraping HTML content from a page, and parsing HTML code with Beautiful Soup.

For the collaborative recommender system, we learned from Luo's article (Luo, 2018). The article mentions collaborative filtering approaches within the nearest neighborhood and matrix factorization using the formulas presented to get cosine similarity and make a prediction for the target user.

The selection of similarity metrics for collaborative filtering was learned from an article from towardsdatascience.com (Saluja, 2018). This article mentioned the difference between Pearson, cosine, euclidean, and adjusted cosine similarity. Use Pearson when the data is subject to user bias/ different ratings scales. Cosine similarity is used when the data is sparse, which means many ratings are undefined. For Euclidean, if the data is not sparse and the magnitude of the attribute values is significant. Finally, use adjusted cosine for the item-based approach (which we will not consider for our project dataset) to adjust for user bias.

Moreover, the article from towardsdatascience.com (Gupta, 2018) mentions the difference between Jaccard Index and Cosine Similarity, where to use each method, and the pros and cons. Briefly, Jaccard Similarity or intersection over union is defined as the size of the intersection divided by the size of the union of two sets, and cosine similarity calculates similarity by measuring. Therefore, it is helpful for the model to decide a more accurate distance between two matrices for different dataset types.

Rating-based recommender system was learned from an article from towardsdatascience.com (Sathwick, 2020). It is a method using correlation instead of cosine similarity to recommend movies. Movies with higher ratings and several reviews will be recommended.

A paper mentioned a recommender system based on sentiment analysis in 2021 (Asani et al., 2021). The author mentioned that sentiment analysis has become popular in designing recommender systems in various fields like restaurant and food areas. Analyzing users' opinions and the extraction of preferences from reviews will lead to a personalized recommender system. Similarly, we applied this idea in IMDb and implemented a series of models to build personalized movie recommenders.

DATA

Three datasets will be introduced in this project. The first one is the IMDB Dataset of 50K Movie Reviews, which is collected from Kaggle. 50,000 rows and two columns are included in this dataset. Each row has a text comment of a movie and a sentiment label for the review, which is shown in figure 1.

	review	sentiment
0	One of the other reviewers has mentioned that ...	positive
1	A wonderful little production. The...	positive
2	I thought this was a wonderful way to spend ti...	positive
3	Basically there's a family where a little boy ...	negative
4	Petter Mattei's "Love in the Time of Money" is...	positive

Figure 1. Screenshot of the dataset

Then, the length of each review was calculated. Figure 2 demonstrates the statistical summary, and figure 3 shows the distribution of review length, respectively. From the results, we can see that the minimum length includes 32 words. In contrast, the most extended review contains 13,704 words. Furthermore, the distribution of review length is right-skewed. Most reviews come with content between 32 and 2,000 words. Figure 4 is the review length comparison between positive and negative. It displays that the length of most positive reviews is longer than negative reviews. Besides, there are no missing values in this dataset. However, 481 duplicated rows. By calculating the count of positive and negative recordings in this dataset, we believe the purpose of duplication is oversampling. Hence, duplicated rows remained.

```
count    50000.000000
mean      1309.431020
std        989.728014
min         32.000000
25%        699.000000
50%        970.000000
75%       1590.250000
max      13704.000000
Name: text_length, dtype: float64
```

Figure 2. Statistical summary of review length

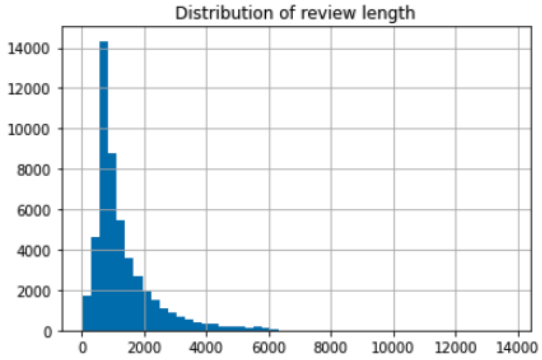

Figure 3. Histogram of review length

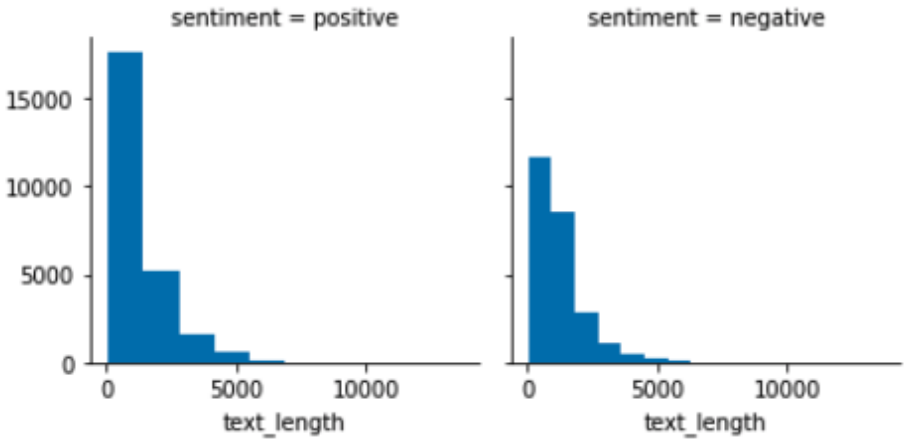

Figure 4. Review length comparison between positive and negative

The second dataset was crawled from IMDb. Due to the limited computation ability of the laptop, this project will focus on one playlist. This playlist describes the top 250 popular movies from IMDb. Information and reviews of the 250 movies will be extracted to determine the optimal movie within the playlist and recommend it to users. Beautiful Soup is applied to parse movie information from IMDb's HTML document.

After that, the second dataset is collected, with 250 rows and six columns. Included features are "movie title," "introduction," "genre," "rating," "release year," and "total number of reviews" (Figure 5). Figure 6 and 7 demonstrates the word cloud of the movie title. "Ring," "Man," "Star," and "War" are popular words within the movie title. Similarly, "Young," "two," "year," and "life" are the most frequent words in movie introduction. From figure 8, which presents the distribution of review numbers, we can see that a large portion of movies has received less than 2,000 reviews, and the most significant review number is more than 10,000. In addition, 8.2 is the common rating for movies in this playlist, and the highest score is 9.2. Figure 10 is the bar chart of the movie genre, and it shows that "Drama" is the most popular genre in this dataset. Lastly, the bar chart of the released year (Figure 11) states that the most popular movies were created within the last thirty years. Furthermore, in this popular movie list, the oldest movie is released in the 1920s.

	Movie_title	Movie_intro	Genre	rating	release_year	review_num
0	Spider-Man: No Way Home	With Spider-Man's identity now revealed, Peter...	[Action, Adventure, Fantasy]	8.7	2021	5000
1	Dune	Feature adaptation of Frank Herbert's science ...	[Action, Adventure, Drama]	8.1	2021	5100
2	The Shawshank Redemption	Two imprisoned men bond over a number of years...	[Drama]	9.3	1994	9900
3	The Dark Knight	When the menace known as the Joker wreaks havo...	[Action, Crime, Drama]	9.0	2008	7900
4	Jurassic Park	A pragmatic paleontologist touring an almost c...	[Action, Adventure, Sci-Fi]	8.1	1993	1300

Figure 5. Screenshot of movie information dataset

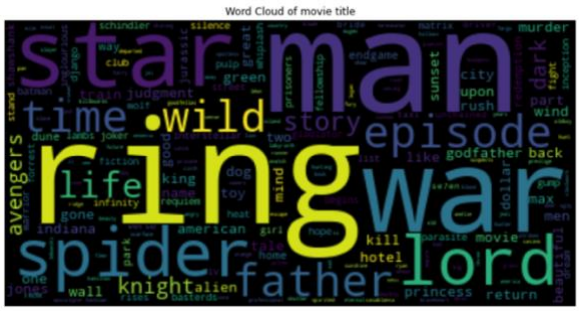

Figure 6. Word cloud of movie title

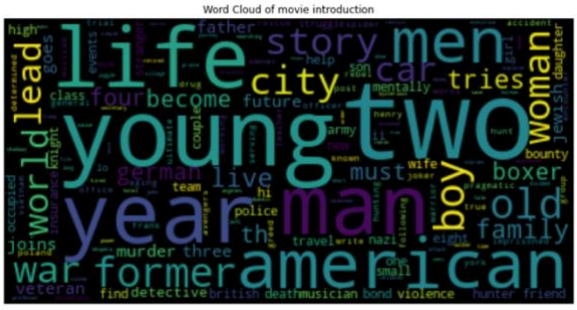

Figure 7. Word cloud of movie introduction

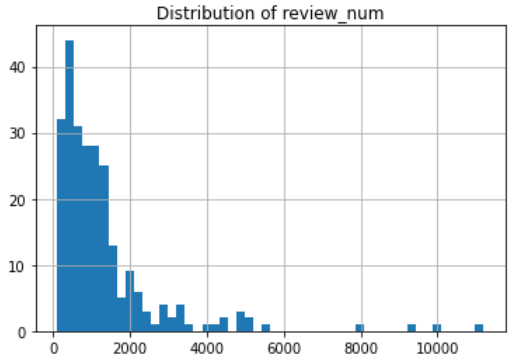

Figure 8. Distribution of review numbers

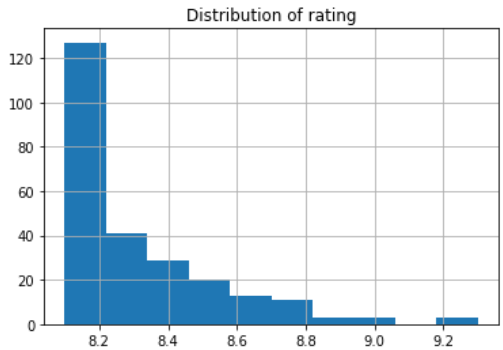

Figure 9. Distribution of rating

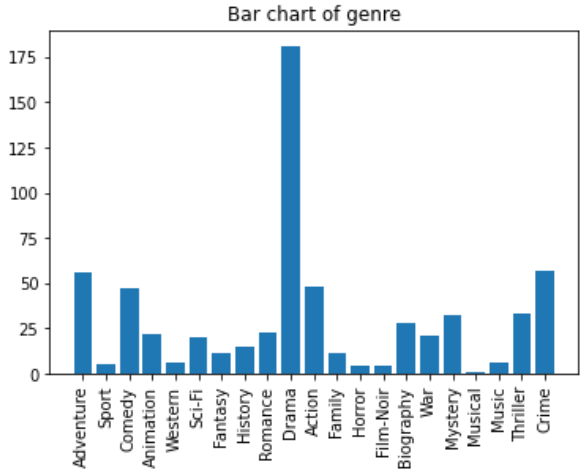

Figure 10. Bar chart of movie genre

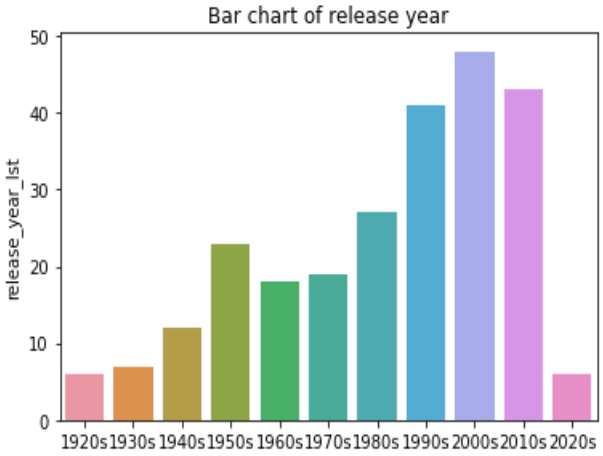

Figure 11. Bar chart of released year

The third dataset for this project is crawled from the IMDB website includes 272,863 reviews and six features. The features include "movie title," "user name," "rating score," "num helpful," and "review" (Figure 12). Figure 13 demonstrates the word cloud of reviews "great," "first," "best," "good," "watched," "like." There are 161,008 unique users and 247 unique movie titles (Figure 16). For the "review" column, 206,393 reviews are positive, and 55,469 reviews are negative (Figure 14). The earliest review was posted on 1998-07-29, and the latest review was posted on 2022-02-16 (Figure 15).

	Movie_title	user_name	rating_score	rating_time	num_helpful	review
1	The Shawshank Redemption	TheLittleSongbird	10	2009-04-17	0.75	Shawshank Redemption is without doubt one of t...
2	The Shawshank Redemption	bkoganbing	9	2011-02-17	0.72	None of the usual otherworld creatures that po...
3	The Shawshank Redemption	Leofwine_draca	10	2016-12-18	0.81	Based on a novella by Stephen King, this is be...

Figure 12. Screenshot of review dataset

Figure 13. Word cloud of movie review

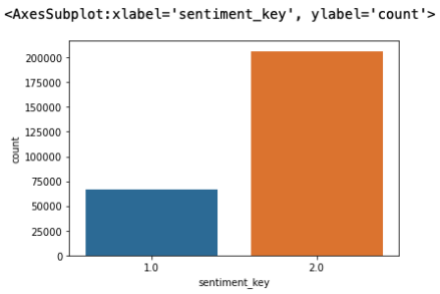

Figure 14. # Of positive/negative review

```
In [58]: reviews_250['rating_time'].min()
Out[58]: '1998-07-29'

In [59]: reviews_250['rating_time'].max()
Out[59]: '2022-02-16'
```

Figure 15. earliest/latest review posted

```
In [72]: reviews_250['user_name'].nunique()
Out[72]: 161008

In [73]: reviews_250['Movie_title'].nunique()
Out[73]: 247
```

Figure 16. # Of unique user/movie title

METHODS

Sentiment analysis includes eight models - *DNN*, *CNN-LSTM*, *TFIDF with Logistic Regression*, *Linear SVM*, *Multinomial Naïve Bayes*, and *BOW with Logistic Regression*, *Linear SVM*, *Multinomial Naïve Bayes*. After implementing multiple models, the best models were selected based on accuracy and Area under the ROC Curve (AUC).

Web crawler used the *BeautifulSoup* method.

Recommender systems were implemented using three methods - *collaborative filtering-based recommender*, *content-based recommender*, and *rating-based recommender*.

Sentiment Analysis

Following are the details of sentiment analysis.

GloVe + DNN / CNN-LSTM

In the beginning, reviews were preprocessed by removing special characters, replacing punctuation, dropping stop words, and extracting the tokens from the string. Glove - an unsupervised learning algorithm was introduced to obtain vector representations for words. After obtaining the vector of each token, we average the value of all vectors to represent each review. Also, a simple DNN model and a hybrid CNN-LSTM model were presented. Figure 17 and figure 18 are the structure of the DNN model and a hybrid CNN-LSTM model, respectively.

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 200)	20200
dropout (Dropout)	(None, 200)	0
dense_1 (Dense)	(None, 100)	20100
dropout_1 (Dropout)	(None, 100)	0
dense_2 (Dense)	(None, 100)	10100
dropout_2 (Dropout)	(None, 100)	0
dense_3 (Dense)	(None, 1)	101
activation (Activation)	(None, 1)	0
dense_4 (Dense)	(None, 1)	2

Total params: 50,503
Trainable params: 50,503
Non-trainable params: 0

Figure 17. Structure of DNN model

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 100, 32)	128
max_pooling1d (MaxPooling1D)	(None, 50, 32)	0
conv1d_1 (Conv1D)	(None, 50, 32)	3104
max_pooling1d_1 (MaxPooling1D)	(None, 25, 32)	0
lstm (LSTM)	(None, 100)	53200
dense_5 (Dense)	(None, 1)	101

Total params: 56,533
Trainable params: 56,533
Non-trainable params: 0

Figure 18. Structure of hybrid CNN-LSTM model

TFIDF + Logistic Regression / Linear SVM / Multinomial Naïve Bayes

TFIDF is a technique to quantify words in documents, and it is a widely used technique in information retrieval and text mining. It will compute a score for each word to represent its importance in the document. By vectorizing the documents, the programming language will be able to understand textual data in the form of numerical values. The formula for TFIDF is –

TF-IDF = Term Frequency (TF) * Inverse Document Frequency (IDF)

Term frequency measures the frequency of a word in a document; a document frequency measures the importance of documents in a whole corpus set (the entire document set). Before implementing TFIDF, data preprocessing includes text normalization, removing HTML strips and noise text, removing special characters, text stemming, removing stop words, and normalizing train/test reviews. After implementing TFIDF, logistic regression, linear SVM, and

multinomial Naïve Bayes were implemented so that we could have multiple metrics to compare the performance of different models.

BOW + Logistic Regression / Linear SVM / Multinomial Naïve Bayes

Similar to TFIDF, BOW is also a way of representing text data by extracting features from the text when modeling text with machine learning algorithms. The data preprocessing part is the same as TFIDF data preprocessing. Processed data were input into BOW, and similarly, logistic regression, linear SVM, and multinomial Naïve Bayes were implemented.

Model	**Accuracy**	**AUC**
GloVe + DNN	0.8024	0.8819
GloVe + CNN-LSTM	0.7827	0.8694
TFIDF + Logistic Regression	0.7498	0.75
TFIDF + Linear SVM	0.51	0.51
TFIDF + Multinomial Naïve Bayes	0.75	0.75
BOW + Logistic Regression	0.751	0.75
BOW + Linear SVM	0.58	0.58
BOW + Multinomial Naïve Bayes	0.75	0.75

Based on the summary table, we will use *GloVe + DNN* for the sentiment analysis due to its best performance across all models.

Web Crawler

BeautifulSoup

To get the information we need for stage 3, the recommender system, we first need to understand the HTML structure IMDB website and decipher data encoded in URLs, and then we use requests and Beautiful Soup for scraping and parsing data from the web. By using the steps mentioned above, we crawled the playlist page to store 250 movies in a dictionary: titles as the key and each movie's sublink as the value:

```
movie_dic
```

```
{'The Batman': '/title/tt1877830/',
 'Spider-Man: No Way Home': '/title/tt10872600/',
 'The Dark Knight': '/title/tt0468569/',
 'Dune': '/title/tt1160419/',
 'Batman Begins': '/title/tt0372784/',
 'The Dark Knight Rises': '/title/tt1345836/',
 'The Shawshank Redemption': '/title/tt0111161/',
 'Joker': '/title/tt7286456/',
 'The Godfather': '/title/tt0068646/',
```

Figure 19. sample output for movie dictionary

Then, after understanding the web structure, we used each movie's sublink to get all user review information, including details like "user_name," "rating_time," "num_helpful," and "review

text." We saved this information set as a dictionary for later collaborative filtering-based recommender use.

	Movie_title	user_name	rating_score	rating_time		num_helpful	review
0	0	0	0	0	0	0.00	0
0	Last Survivors	siderite	5	2022-02-13	0	1.00	It's difficult to talk about the film without ...
1	Last Survivors	stevendbeard	6	2022-02-07	1	0.20	I saw Last Survivors, starring Drew Van Acker-...
2	Last Survivors	kwenchow	1	2022-02-14	2	0.00	This film start with a man doing his daily rou...
3	Last Survivors	radhrh	2	2022-02-05	3	0.61	While some of the cinematography of the frozen...
4	Last Survivors	tkarlmann	8	2022-02-05	4	0.42	Acting here was excellent -- so good that I ju...
5	Last Survivors	legionofthesnowzombie	4	2022-02-06	5	0.71	The story would have made a good short, say 45...
6	Last Survivors	pipo-	3	2022-02-05	6	0.75	It started off good with premise of wars and h...

Figure 20. sample output of review dataset

We applied the same logic to scrape each movie' sublink to get movies' feature for the content-based recommender. The dictionary includes "movie introduction," "genre," "rating," "release year," and "review number."

Recommender System

Four recommender systems were implemented as follows.

Collaborative Filtering Based Recommender

Collaborative Filtering is to give recommendations to a user based on preferences of "similar" users. There are two categories of Neighborhood-Based models for collaborative recommenders: User-Based and Item-based Collaborative Filtering. We will mainly use User-Based for this project. The user-Based model is that for a target user u, find the top K most similar users based on their rating profiles who have rated the target item by using K-Nearest-Neighbor (KNN) strategy; then predicted rating on target item is the weighted average of ratings by the neighbors.

In the collaborative filtering matrix, we usually assume that features in the data are similar objects, the object is movie names in our dataset, and usually requires "explicit" ratings of objects by users based on a rating scale. The collaborative filtering process uses the nearest neighbor strategy and then makes predictions. For the K-nearest-neighbor (KNN) strategy, the basic idea is to find other users with the most similar preferences or tastes to the target user, which needs a matrix to compute similarities among users based on their ratings of items. Since we will use 0 to present those scores that users have not rated for some movies, our data is sparse (many ratings are undefined). Therefore, we will use cosine similarity based on the degree of correlation between user X and user Y:

$$sim(X,Y) = \frac{\sum_i (x_i \times y_i)}{\sqrt{\sum_i {x_i}^2 \times \sum_i {y_i}^2}}$$

In this case, 1 means very similar, 0 means no correlation, and -1 means dissimilar. Then for making prediction part, when generating predictions from the nearest neighbor, neighbors can be weighted based on their distance to the target user. For example, to generate predictions for a target user X on item i:

$$P_{X,i} = \bar{r_X} + \frac{\sum_{Y=1}^{k} (r_{Y,i} - \bar{r_Y}) \times sim(X,Y)}{\sum_{Y=1}^{k} sim(X,Y)}$$

$$\bar{r_X} = mean\ rating\ for\ user\ X$$

$$Y_1, \dots Y_k\ are\ the\ k-nearest-neighbors\ to\ X$$

$$r_{Y,i} = rating\ of\ user\ Y\ on\ item\ I$$

$$sim(X,Y) = cosine\ similarity\ between\ X\ and\ Y$$

The result is a weighted average of deviations from the neighbors' mean ratings, and closer neighbors count more. Next, we use a for loop to get all the predicted scores for all the movies that the target user has not rated, then sort the scores and find top K movies to recommend to the target user.

Initially, we used the most common way to calculate the user's rating score matrix to get the similarities with cosine similarity in the Pandas DataFrame line by line, sort the DataFrame by correlation value, and then get prediction scores for each movie one by one, and finally return the top K items for the target user. However, the running time is vast due to our enormous review data size. It took four fours to finish running the entire review dataset.
Apparently, the straightforward way is not suitable for this dataset; therefore, we made slight changes to improve the running time in a few places. First, we used SET{} to store the data locally for further use to save time and save running space. The SET{} stores all the review scores for all movies and use 0 to present as no rating score for this movie (Figure 21). After using the SET{}, the model only takes about 1 minute to finish running on the review dataset. Compared to the original running time, it is a considerable improvement.

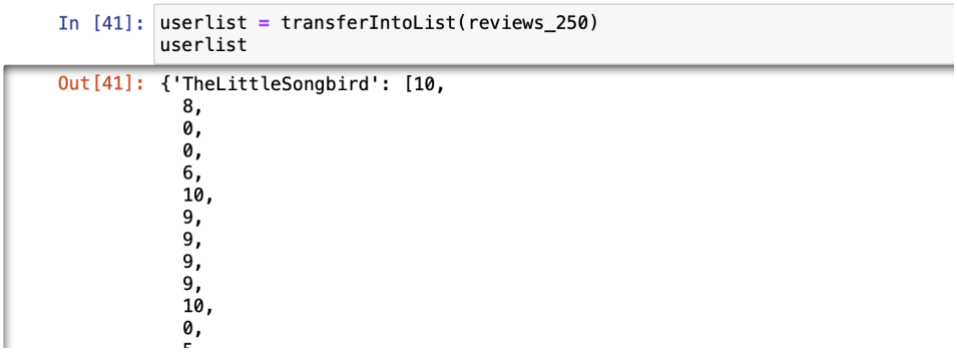

Figure 21. Example of SET store review table

We typed “The Dark Knight” as the example that assumes the target user only rated this movie as 10. We applied the collaborative model, and the system recommends ten movies that same taste users rated high scores, as shown below (Figure 22).

```
K = 10
collaborative(numlist, K, userlist)
```

```
Good Will Hunting
Logan
Interstellar
Gone Girl
Whiplash
Mad Max: Fury Road
The Lion King
The Shawshank Redemption
The Wolf of Wall Street
Avengers: Endgame
```

Figure 22. Example of Collaborative Recommender

Collaborative Filtering Based Recommender + Sentiment Analysis
After running the collaborative-filtering-based recommender on each user's number scores on movies ranging from 0-10, we applied sentiment analysis on the review column that we crawled from the IMDB website. Converted reviews into positive or negative, using 1 and 2 to present (1: negative, 2: positive). Same as the previous model, we still utilize SET{} to store the review

dataset. Shown below are the recommended outputs of an example of target user rated “The Dark Knight” as a positive review which is 2 (Figure 23).

```
K = 10
collaborative(numlist, K, userlist_1)
```

```
Interstellar
Gone Girl
Inside Out
Whiplash
Mad Max: Fury Road
The Lion King
The Shawshank Redemption
Dead Poets Society
The Wolf of Wall Street
The Godfather
```

Figure 23. Example of Collaborative Recommender + sentiment analysis

Content Based Recommender

Content-based recommender uses item features to find similar content that users like based on their previous performance or explicit feedback. The movie information dataset includes "movie title," "introduction," "genre," "rating," "release year," and "total number of reviews." Since features "Movie_title" and "Movie_intro" are string types, GloVe was implemented to convert text-based features to vectors. The dataset ends with a size of 250 rows and 224 columns. Next, a list of movie genres will be presented, and users will be inquired about their favorite genre of movie. Then, list all movie titles with the same category will be demonstrated. The same as the previous step, users are requested to name their favorite movie. Then by calculating the cosine similarity between the favorite and other movies, the top 10 most similar ones will be recommended to the customer.

Figure 24 explains the steps of conducting the content-based recommender. The user starts with choosing "Action" as their favorite one from the list of genre names. As a result, "Spider-man," "Dune," "The Dark Knight," and other action movies are given as options for the user to decide further. Subsequently, the user chose "The Dark Knight" as their favorite movie. Eventually, "Avengers," "The Shawshank," "Joker," and other movies were recommended since they have the closest cosine similarity with "The Dark Knight."

```
'Action','Adventure','Animation', 'Biography', 'Comedy', 'Crime', 'Drama', 'Family', 'Fantasy', 'Film-Noir',
Which is your favorite genre of movie?
 Action

['Spider-Man: No Way Home', 'Dune', 'The Dark Knight', 'Jurassic Park', 'Avengers: Endgame', 'The Lord of th
Which is your favorite movie?
The Dark Knight

Here is the recommendation for you:

Avengers: Endgame
The Shawshank Redemption
Joker
The Lord of the Rings: The Fellowship of the Ring
Dune
Spider-Man: No Way Home
The Godfather
Interstellar
The Matrix
Inception
```

Figure 24. Example of Content Based Recommender

Rating Based Recommender

Rating-based recommender works by comparing the one movie ratings given by all users with another movie ratings given by all users. Instead of using similarity, the correlation between

movies from all user's perspectives was compared. This method is only used to move titles and ratings from the crawled data.

First, we put the movie names and the average ratings of each movie into a dataset. Then, we counted the number of ratings for each movie. Next, we created a table with the rows as the user names and the columns as movie titles. The values of the table represent the rating for each movie by every user. Based on this table, we can find the correlation between users who watched the movie with remaining all other movies. Finally, we sorted the results in descending order and recommended them to users.

The screenshot below shows that when entering "The Dark Knight" as input, a series of movies will be recommended to the user.

```
rs = RecommendationSystem()
rs.recommend(Movie_title= 'The Dark Knight')
```

```
Skipping line 166960: unexpected end of data

Movies with similar ratings like your input movie ---> The Dark Knight <--- are

The Departed
The Dark Knight Rises
Batman Begins
Gone Girl
The Grand Budapest Hotel
Inception
The Prestige
Prisoners
Interstellar
Whiplash
```

Figure 25. Example of Rating Based Recommender

CONCLUSIONS

	Collaborative Filtering Based Recommender	Collaborative Filtering Based Recommender + Sentiment Analysis	Content Based Recommender	Rating Based Recommender
Recommended Movies	Good Will Hunting	Interstellar	Avengers: Endgame	The Departed
	Logan	Gone Girl	The Shawshank Redemption	The Dark Knight Rises
	Interstellar	Inside Out	Joker	Batman Begins
	Gone Girl	Whiplash	The Lord of the Rings: The Fellowship of the Ring	Gone Girl
	Whiplash	Mad Max: Fury Road	Dune	The Grand Budapest
	Mad Max: Fury Road	The Lion King	Spider-Man: No Way Home	Hotel
	The Lion King	The Shawshank Redemption	The Godfather	Inception
	The Shawshank Redemption	Dead Poets Society	Interstellar	The Prestige
	The Wolf of Wall Street	The Wolf of Wall Street	The Matrix	Interstellar
	Avengers: Endgame	The Godfather	Inception	Whiplash

Figure 26. Recommendations from 4 models

An experiment for running our Recommendations with a target user with a favorite movie, "The Dark Knight." Due to the recommender system being unsupervised learning and cannot compare the accuracy of four models, we would like to study those most selected movies. "Interstellar" is the optimal movie that four models recommend. Following badge, films happen in 3 lists are "Gone Girl," "Whiplash," and "The Shawshank Redemption." Moreover, "Avengers," "Spider-

Man," and "Logan" appear in the list – all superheroes' themes. In addition, "Batman Begins," "The Dark Knight," and "The Dark Knight Rises" of Batman Trilogy are all identified too. Next, both "The Dark Knight" and "Joker" movies presented with the same character – "Joker," and most people believe they are connected. Our models recognized all these movies, which demonstrates that the Recommendation strategy is effective and reasonable. Finally, we would like to present both the most voted movies and the complete list. The result would explain to users the reason behind the recommendation, whether it is determined by browsing history or the movie they like. We believe this recommender system can navigate users to the right movies that they are interested in and willing to watch them.

ACKNOWLEDGMENTS

We would like to acknowledge the help and support from Dr. Ilyas Ustun through the quarter.

AUTHOR CONTRIBUTIONS

The authors confirm contribution to the paper as follows: study conception and design: Xiaojing Shen, Serena Yang, Yue Hou; data collection: Xiaojing Shen, Serena Yang; analysis and interpretation of results Xiaojing Shen, Serena Yang, Yue Hou; draft manuscript preparation: Xiaojing Shen, Serena Yang, Yue Hou. All authors reviewed the results and approved the final version of the manuscript.

REFERENCES

1. Rehman, Malik, A. K., Raza, B., & Ali, W. (2019). A Hybrid CNN-LSTM Model for Improving Accuracy of Movie Reviews Sentiment Analysis. *Multimedia Tools and Applications*, 78(18), 26597–26613. Retrieved March 14, 2022, from https://doi.org/10.1007/s11042-019-07788-7

2. Gomez-Uribe, C. A., & Hunt, N. (2015). The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4), 1-19. Retrieved March 14, 2022, from https://www.semanticscholar.org/paper/The-Netflix-Recommender-System-Gomez-Uribe-Hunt/e9dd899f0e599eafb4fe47696c83d07d971c0088

3. William, S. (2021). TF-IDF for document ranking from scratch in Python on Real World Dataset. *Medium.* Retrieved March 14, 2022, from https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089

4. Sathwick, R. (2020). Implementing a recommendation system on IMDB dataset through Machine Learning Techniques. *Medium.* Retrieved March 14, 2022, from https://medium.com/@sr7037/implementing-a-recommendation-system-on-imdb-dataset-through-machine-learning-techniques-47d0a86da9df

5. Asani, E., Vahdat-Nejad, H., & Sadri, J. (2021). Restaurant Recommender System based on sentiment analysis. *Machine Learning with Applications*. Retrieved March 14, 2022, from https://www.sciencedirect.com/science/article/pii/S2666827021000578#:~:text=To%20obtain%20the%20opinion%20of,food%20preferences%20of%20the%20user.

6. Sabarman, J. (2020). Selenium Python—Movie Recommendation From IMDB Reviews (without actually read the reviews). *Medium.* Retrieved March 14, 2020, from https://medium.com/analytics-vidhya/movie-recommendation-from-imdb-reviews-without-**actually-read**-the-reviews-fe8865a70bd5.

7. Saluja, C. (2018). Collaborative Filtering based Recommendation Systems exemplified... *Towards Data Science*. Retrieved March 14, 2022, from https://towardsdatascience.com/collaborative-filtering-based-recommendation-systems-exemplified-ecbffe1c20b1

8. Gupta, S. (2018). Overview of Text Similarity Metrics in Python. *Towards Data Science*. Retrieved March 14, 2022, from https://towardsdatascience.com/overview-of-text-similarity-metrics-3397c4601f50

9. Luo, S. (2018). Introduction to Recommender System. *Towards Data Science.* Retrieved March 14, 2022, from https://towardsdatascience.com/intro-to-recommender-system-collaborative-filtering-64a238194a26

10. Breuss, M. (2021). Beautiful Soup: Build a Web Scraper with Python. *Real Python.* Retrieved March 14, 2022, from https://realpython.com/beautiful-soup-web-scraper-python/