BIA-660A

Movie Recommendation System Based On Post-Viewing Reviews

Instructor: Rong Liu

Tze-How Lee, Hao-Chen Wang, Aiwu Song, Xinze Yu

1. Motivation and Research Question

Motivation:

"Every time I go to a movie, it's magic, no matter what the movie's about." – Steven Spielberg.

Everyone loves movies irrespective of age, gender, race, color, or geographical location. We are all connected to each other via this fantastic medium. Yet the most exciting fact is how unique our choices and combinations are in terms of movie preferences. Some people like genre-specific movies, be it a thriller, romance, or sci-fi, while others focus on lead actors and directors. When we take all that into account, it's astoundingly difficult to generalize a movie and say that everyone would like it. But with all that said, it is still seen that a specific part of society likes a similar film.

Therefore we want to apply our method learned in class to build a movie recommendation system.

Research Question:

Are there any common characteristics of popular and highly rated movies from the perspective of review analysis? What are the common reviews people comment on after watching a movie?

We expect this work can make the following contributions:

- We create our own movie critics datasets and cluster movies based on the reviews.
- We provide a recommendation system to help users hop into the next movie journey based on their preferences and the expectations of the emotions or inspiration they want.
- This project can give us a glimpse of popular movies, their characteristics, and the inspiration they bring to people after watching them.

2. Background and Related Work

Machine learning algorithms in recommender systems typically fit into two categories: content-based systems and collaborative filtering systems. Modern recommender systems combine both approaches. Let's take a closer look at both categories.

A) Content-Based Movie Recommendation Systems

Content-based methods are based on the similarity of movie attributes. Using this type of recommender system, if a user watches one movie, similar movies are recommended. For example, if a user watches a comedy movie starring Adam Sandler, the system will recommend them movies in the same genre or starring the same actor, or both. With this in mind, the input for building a content-based recommender system is movie attributes.

B) Collaborative Filtering Movie Recommendation Systems

With collaborative filtering, the system is based on past interactions between users and movies. With this in mind, the input for a collaborative filtering system is made up of past data of user interactions with the movies they watch.

3. Crawling Data & Dataset Description

Crawling Data:

First, we visit the Top 250 movies web page on IMDB (https://www.imdb.com/chart/top), there are hyperlinks and movie names of 250 movies on this interface. So our first step is to define a function to crawl the movie's id, name, and rating and generate links to the movie details page and movie review page.

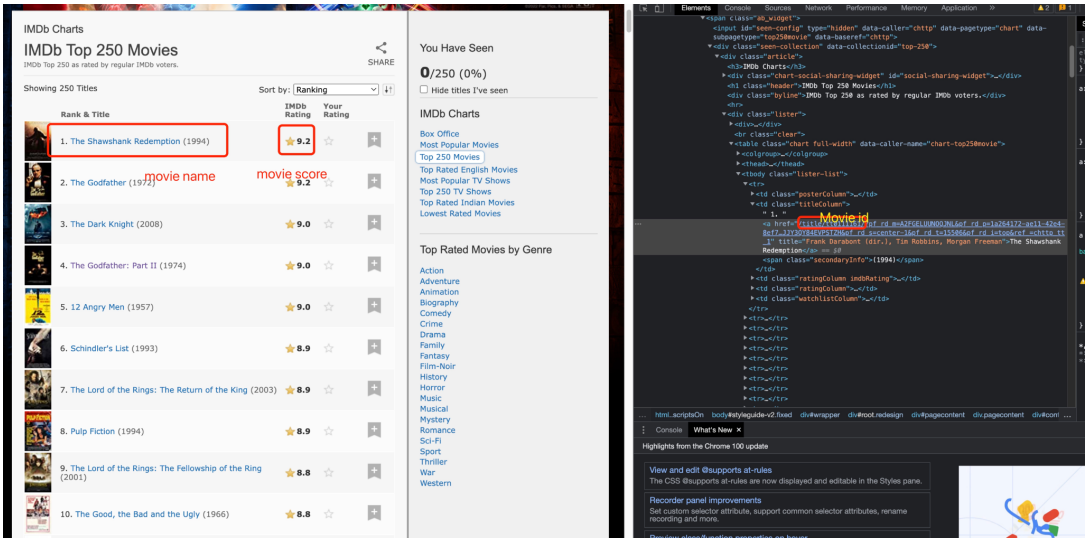

Figure 1 IMDB Website Top 250 Movies

Then, we crawled the first 100 reviews of each movie (sorted by the "Helpfulness").

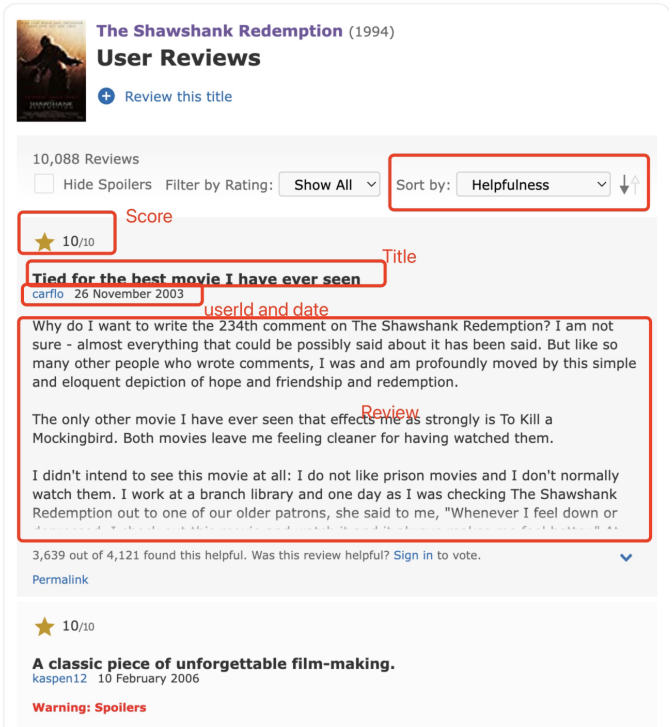

Figure 2 Review of the Movie

Loading comments on the desired comment page require clicking the load more button to do so. Since we crawl 100 reviews for each movie this time, we only need to click three times to load, and then crawl the reviews on the page, and we also crawl the title, user ID, and date of the

review. For some movies with less than 100 reviews, if the load more button is not visible, we stop clicking immediately.

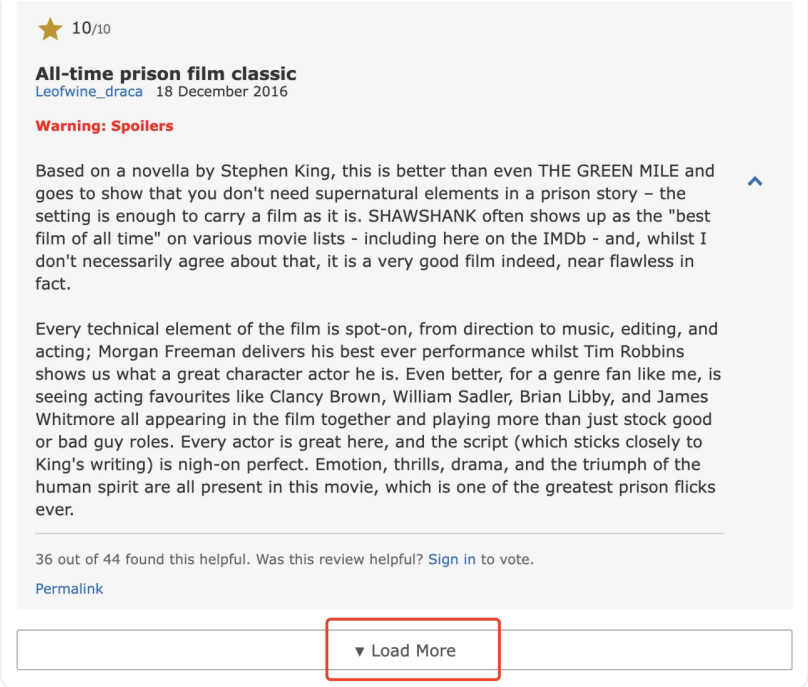

Figure 3 “Load More” Button

Dataset Description:

Our dataset has two major CSV: A list of top 250 movies and a list of the first 100 reviews of these movies:

- Top_movie_list: movieId, movieName, movieScore, movieLink, movieReviewLink
- Movie_review_info: userId, reviewDate, reviewScore, reviewTitle, userReview, movieId

Top_movie_list (Top 250 movies):

	movieId	movieName	movieScore	movieLink	movieReviewLink
0	0111161	The Shawshank Redemption	9.240109237945433	https://www.imdb.com/title/0111161/?ref_=tt_urv	https://www.imdb.com/title/tt0111161/reviews?s...
1	0068646	The Godfather	9.161289887364756	https://www.imdb.com/title/0068646/?ref_=tt_urv	https://www.imdb.com/title/tt0068646/reviews?s...
2	0468569	The Dark Knight	8.99457239775514	https://www.imdb.com/title/0468569/?ref_=tt_urv	https://www.imdb.com/title/tt0468569/reviews?s...
3	0071562	The Godfather: Part II	8.99033695773171	https://www.imdb.com/title/0071562/?ref_=tt_urv	https://www.imdb.com/title/tt0071562/reviews?s...
4	0050083	12 Angry Men	8.950992808370717	https://www.imdb.com/title/0050083/?ref_=tt_urv	https://www.imdb.com/title/tt0050083/reviews?s...

Figure 4 Top 250 Movies List Dataframe

Movie_review_info (Raw Data):

	userId	reviewDate	reviewScore	reviewTitle	userReview	movieId
0	2509775	26 November 2003	[10]	[<a class="title" href="/review/rw0349418/?ref...	[<div class="text show-more__control clickable...	111161
1	1898687	10 February 2006	[10]	[<a class="title" href="/review/rw1288098/?ref...	[In its O...	111161
2	16161013	24 July 2010	[10]	[<a class="title" href="/review/rw2284594/?ref...	[<div class="text show-more__control clickable...	111161
3	1005460	8 February 2001	[]	[<a class="title" href="/review/rw0348718/?ref...	[I have n...	111161
4	997166	27 August 2002	[10]	[<a class="title" href="/review/rw0349147/?ref...	[<div class="text show-more__control clickable...	111161
5	131017397	2 April 2021	[9]	[<a class="title" href="/review/rw6770639/?ref...	[If you l...	111161
6	265899	21 November 2005	[10]	[<a class="title" href="/review/rw1221355/?ref...	[<div class="text show-more__control clickable...	111161
7	16117882	18 February 2008	[10]	[<a class="title" href="/review/rw1822343/?ref...	[<div class="text show-more__control clickable...	111161
8	129753872	23 February 2021	[9]	[<a class="title" href="/review/rw6627363/?ref...	[You have...	111161
9	146338622	4 December 2021	[9]	[<a class="title" href="/review/rw7613597/?ref...	[This is ...	111161

Figure 5 Raw Data Dataframe

4. Dataset Preprocessing & Description

Preprocessing:

Step 1: Using regular expression to extract useful information from messy Html data, turn it into a clean and readable information

Step 2: Drop rows or columns with NaN value of more than 30%

Step 3: Drop duplicate data

Step 4: Missing value: Propagate the last valid observation forward to the next valid backfill

Step 5: Create a new column named“ adjectives_merge” which extract all the adjective from “userReview”, then create “a_length” to measure the adjective word length, if the reviews are less than three words, drop this row to reduce the error.

Step 6: Create new columns called “adjectives_set” and “ adjectives_with_drop_merge” adjectives_set drops all the duplicate adjectives and adjectives_with_drop_merge also drops stopwords for our comparison.

Step 7: Create a new column called label using get_emtion package to get the labels.

Description:

The final dataset for clustering:

	adjectives_merge
16276	British in-between former tragic alcohol-drive...
1683	serial negative directorial dark clinically ex...
23048	hilarious graphic excessive hysterical sacrile...
5694	many excellent perfectionist same nervous impo...
12505	sure perfect fantastic perfect beautiful emoti...
...	...
12889	great many last only slight obvious different ...
22942	soft many amazing many phenomenal strong fit l...
5870	great historical second great wartime first we...
983	remarkable friendly big American congressional...
16893	interesting worth allow high true true similar...

20474 rows × 1 columns

Figure 6 Adjectives_merge Dataframe

	adjectives_merge	label
21336	outstanding true digital investigative remarka...	Happy
12187	commented easy German 'evil Little little Germ...	Angry
12796	Real Real terrible sole simple Wild hot ex-bab...	Sad
14520	Sure outlandishly true adrenaline-pumping firs...	Happy
20162	main real real ethnic little common little oth...	Sad
...	...	...
2398	male abnormal amazing early adoring male human...	Happy
779	outstanding stunning fantastic	Happy
23733	original good same entertaining boring	Happy
301	perfect perfect perfect first much perfect wor...	Sad
23924	amazing Great many magnificent	Happy

2275 rows × 2 columns

Figure 7 Adjectives_merge with Emotion Label

The final dataset for our recommendation system:

	movieId	adj	adjectives_merge	adjectives_set	adjectives_with_drop	adjectives_with_drop_merge	label	movieName
0	12349	[great unsentimental latter soft-shoe physical...	great unsentimental latter soft-shoe physical ...	[analogous, perfect, artistic, evident, hyster...	[analogous, hysterical, artistic, evident, per...	analogous hysterical artistic evident perfect ...	Happy	The Kid
1	15324	[surreal dazzling original perfect dream-like ...	surreal dazzling original perfect dream-like f...	[incorrect, perfect, artistic, favourite, hyst...	[incorrect, perfect, artistic, favourite, hyst...	incorrect perfect artistic favourite hysterica...	Happy	Sherlock Jr.
2	15864	[famous hard young only tough memorable amusin...	famous hard young only tough memorable amusing...	[vivid, unsuccessful, perfect, artistic, evide...	[vivid, unsuccessful, perfect, artistic, evide...	vivid unsuccessful perfect artistic evident im...	Happy	The Gold Rush
3	17136	[social underground dominant upper necessary o...	social underground dominant upper necessary ot...	[structural, minute, indelible, incorrect, cou...	[minute, incorrect, couple, favourite, artisti...	minute incorrect couple favourite artistic liv...	Happy	Metropolis
4	17925	[straightforward clever good valuable sad unbe...	straightforward clever good valuable sad unbel...	[minute, what-do-critics-know, perfect, brief,...	[minute, what-do-critics-know, favourite, perf...	minute what-do-critics-know favourite perfect ...	Happy	The General

Figure 8 Dataset for Recommendation System Dataframe

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 250 entries, 0 to 249
Data columns (total 9 columns):
 #   Column                      Non-Null Count  Dtype 
---  ------                      --------------  ----- 
 0   Unnamed: 0                  250 non-null    int64 
 1   movieId                     250 non-null    int64 
 2   adj                         250 non-null    object
 3   adjectives_merge            250 non-null    object
 4   adjectives_set              250 non-null    object
 5   adjectives_with_drop        250 non-null    object
 6   adjectives_with_drop_merge  250 non-null    object
 7   label                       250 non-null    object
 8   movieName                   250 non-null    object
dtypes: int64(2), object(7)
memory usage: 17.7+ KB
```

Figure 9 Dataset Columns

5. Clustering Model Analysis

We applied the Clustering model (LDA) to filter the useless adjective words in each cluster to improve the evaluation performance and precision of the recommendation system.

First iteration:

```
Topic 0:
[('british', '864.19'), ('old', '856.53'), ('much', '601.04'), ('little', '596.37'), ('young', '591.92'), ('human',
'527.49'), ('real', '500.23'), ('dead', '397.45'), ('self', '389.79'), ('narrative', '382.67'), ('cinematic', '362.9
1'), ('new', '362.14'), ('visual', '346.34'), ('final', '317.85'), ('year', '300.50'), ('non', '254.72'), ('long', '2
39.07'), ('guilty', '229.25'), ('complex', '213.56'), ('american', '208.01')]


Topic 1:
[('great', '5167.56'), ('first', '3311.13'), ('many', '2832.02'), ('perfect', '2487.88'), ('good', '1791.42'), ('beau
tiful', '1738.91'), ('much', '1648.13'), ('wonderful', '1587.09'), ('excellent', '1541.33'), ('brilliant', '1526.7
4'), ('new', '1506.01'), ('original', '1476.04'), ('special', '1444.94'), ('amazing', '1422.65'), ('classic', '1410.6
4'), ('fantastic', '1246.76'), ('incredible', '1208.19'), ('last', '1178.41'), ('memorable', '1054.79'), ('favorite',
'1024.95')]


Topic 2:
[('many', '1538.28'), ('human', '1249.70'), ('real', '1198.68'), ('american', '1040.20'), ('true', '986.46'), ('germa
n', '937.19'), ('powerful', '920.81'), ('french', '811.24'), ('different', '603.68'), ('important', '578.88'), ('grea
t', '542.87'), ('political', '539.24'), ('modern', '467.81'), ('young', '467.65'), ('strong', '443.36'), ('anti', '43
7.19'), ('hard', '416.96'), ('social', '410.99'), ('main', '401.48'), ('realistic', '392.30')]


Topic 3:
[('first', '1766.85'), ('black', '1739.94'), ('many', '1640.01'), ('white', '1540.72'), ('old', '1072.75'), ('young',
'1022.38'), ('little', '963.63'), ('much', '852.96'), ('japanese', '829.07'), ('last', '755.82'), ('long', '707.05'),
('american', '693.94'), ('western', '630.06'), ('good', '593.68'), ('second', '543.09'), ('different', '531.47'), ('b
ig', '522.65'), ('great', '471.58'), ('new', '467.04'), ('several', '462.01')]


Topic 4:
[('good', '8351.93'), ('great', '5465.20'), ('bad', '2851.26'), ('much', '2532.73'), ('many', '2454.68'), ('little',
'2092.69'), ('first', '2081.13'), ('real', '1838.83'), ('whole', '1411.46'), ('old', '1385.06'), ('big', '1222.36'),
('funny', '1220.77'), ('main', '1055.83'), ('sure', '1029.61'), ('hard', '981.61'), ('interesting', '975.36'), ('diff
erent', '897.26'), ('wrong', '875.10'), ('top', '806.93'), ('last', '805.15')]
```

Figure 10 First LDA Result

The drop words are selected by us manually: ['much', 'first', 'many', 'non', 'human', 'last', 'main', 'sure', 'anti', 'top', 'whole']

Then, we repeat the steps for several iterations. In total, 33 frequent but meaningless adjective words have been selected into our stop words list within 6 iterations.

```
drop_lists1 = ['much', 'first', 'many', 'non', 'human', 'last', 'main', 'sure', 'anti', 'top', 'whole']
drop_lists2 = ['little', 'able', 'true', 'hard', 'final', 'full', 'second', 'entire']
drop_lists3 = ['long', 'year', 'big', 'small', 'several']
drop_lists4 = ['short', 'overall', 'next', 'obvious']
drop_lists5 = ['self', 'right', 'actual']
drop_lists6 = ['third', 'personal']
```

Figure 11 Additional Stop Words

6. Recommendation System

With all adjective words collected and flittered from reviews. We concatenated these words into one row for each movie and filtered the redundant words when applying the clustering model, then transformed the rest of the words into a similarity matrix. The Recommendation System is based on a function that relies upon a cosine similarity matrix to generate results. The movie name is the input for the function. As a result, after entering a movie name into this Recommendation System, we can get the most N similar movies, which represent those movies that possess similar emotion types.

```
recommend('Vertigo', 6, cosine_sim_drop)
```

```
['Rebecca',
 'The Secret in Their Eyes',
 'Rear Window',
 'No Country for Old Men',
 'The Prestige',
 'Psycho']
```

Figure 12 Result From Recommendation System

We designed a simple user interface to show the application of the recommendation system.

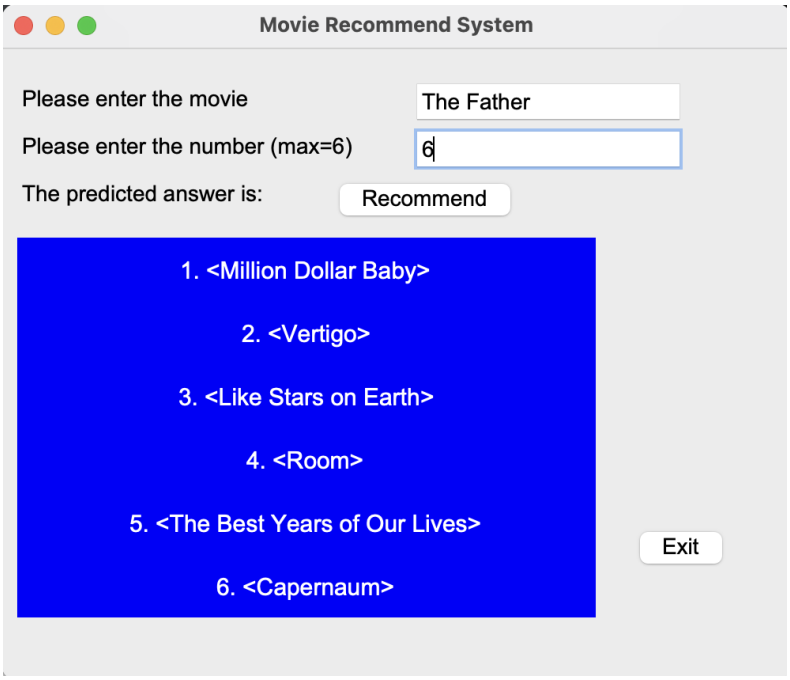

Figure 13 Recommendation System UI

7. Sentiment Analysis

We apply a Python package: Text2emotion, to extract the emotions from users' reviews:

- Text2emotion can process any textual message and recognize the emotions embedded in it.
- It is compatible with 5 different emotion categories: Happy, Angry, Sad, Surprise, and Fear. The output result would present the percentage of 5 types of emotions.
- We determine the emotion type of review by extracting the highest percentage of emotion to be the emotion label.

Afterward, we do the sentiment analysis for adjectives and then plot two emotion distribution histograms:

- Left histogram: the distribution of emotion type from every single review.
- Right histogram: the distribution of emotion type from each movie.

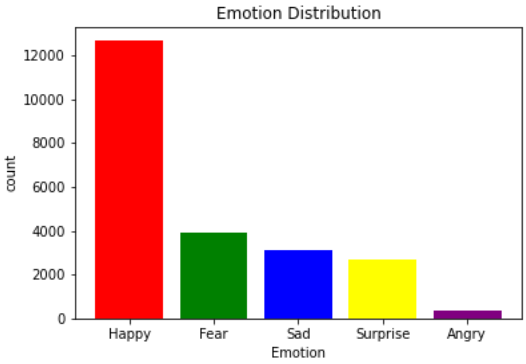

Figure 14 Emotion Distribution of Each Review

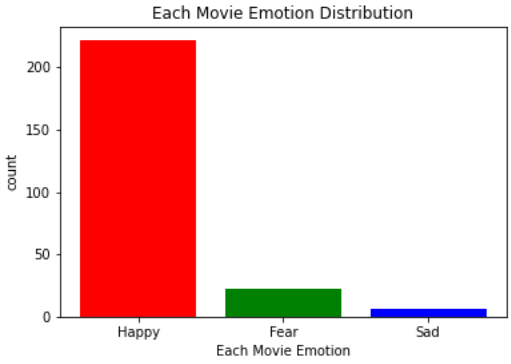

Figure 15 Emotion Distribution of Each Movie

The results show that our database currently has a high percentage of positive comments. The main sentiment is "Happy". In the individual review part, the five emotions of this Python package are all assigned. The difference between the individual review and the overall review of the film lies in the two emotions of surprise and fear. These two emotions can be seen when the individual reviews are analyzed, while the whole reviews of the movie are classified by using the maximum proportion of emotions, these two emotions disappear.

8. LDA & KMeans Model Analysis

We used emotion labels from sentiment analysis to perform two unsupervised clustering models LDA & KMeans. external evaluations and then predict the accuracy of two unsupervised clustering model results.

```
external_evaluate(pred_clusters_lda, test_df['label'])
```

	precision	recall	f1-score	support
Angry	0.00	0.00	0.00	39
Fear	0.00	0.00	0.00	402
Happy	0.55	1.00	0.71	1252
Sad	0.00	0.00	0.00	321
Surprise	0.00	0.00	0.00	261
accuracy			0.55	2275
macro avg	0.11	0.20	0.14	2275
weighted avg	0.30	0.55	0.39	2275

Figure 16 LDA Prediction Result

```
external_evaluate(pred_clusters_kmean, test_df['label'])
```

	precision	recall	f1-score	support
Angry	0.00	0.00	0.00	39
Fear	0.00	0.00	0.00	402
Happy	0.57	0.94	0.71	1252
Sad	0.38	0.26	0.31	321
Surprise	0.00	0.00	0.00	261
accuracy			0.55	2275
macro avg	0.19	0.24	0.20	2275
weighted avg	0.37	0.55	0.43	2275

Figure 17 KMeans Prediction Result

The results show that the predicted results of the two models are not satisfactory, since our data mainly contains positive emotions therefore the results tend to be single. The LDA algorithm tends to predict all the results as Happy, while the KMeans algorithm predicts some results as Sad. Although the accuracy of both is about the same, according to F1-score If we decide to work on future work, the KMeans algorithm is more suitable for our situation since it's more in line with the actual results.

9. Analysis of Experiment Results

Recommendation System

We created our own movie critics dataset and provided a recommendation system to help users hop into the next movie journey based on the reviews. It did work smoothly, but we found several issues that we can improve.

1. It is difficult for us to evaluate the recommendations system results since our results are generated based on the similarity of the filtered reviews. If we're inside a tech company, we have more resources to evaluate the system's effectiveness. Such as user mouse clicks, searches, and viewing history.
2. Our dataset lacks movie genres (tags). Introducing more data sources can help build a better recommendation system.

Models and Analysis

In the analysis part, the two clustering models do not have a good performance. The main reason is the emotions of the comments in our database are mainly “happy“ attributes (Figure 18, Top 20 adjective words from all reviews after filtering stop words), so the results predicted by the model will all tend to predict “happy.” We deem that since the Text2emotion package might not define emotion precisely from the adjective words of reviews. In addition, we only extract one highest emotion from one review, but one review still contains other types of emotion, which may cause the result that some reviews cluster to the wrong groups.

```
[('great', 12701),
 ('good', 11508),
 ('real', 5145),
 ('old', 4017),
 ('young', 3548),
 ('new', 3425),
 ('bad', 3389),
 ('different', 3350),
 ('perfect', 3214),
 ('beautiful', 2765),
 ('classic', 2579),
 ('excellent', 2517),
 ('original', 2445),
 ('brilliant', 2284),
 ('amazing', 2251),
 ('wonderful', 2208),
 ('funny', 2078),
 ('interesting', 2076),
 ('special', 1999),
 ('American', 1979)]
```

Figure 18 Top 20 adjective words

Future Work

We believe that importing the tags of movies (comedy, horror, action…etc.) can allow us to better evaluate the results of the recommendation system. We also need to expand our review database, especially for some movies with both positive and negative reviews. The introduction of more negative reviews can help us build a more accurate model.

Objective Detail
Crawl more movie reviews (positive and negative) from different types of movies.
Crawl movie genres (tags).
Find out a better sentiment analysis method. Maybe create our own model.
Apply and evaluate more clustering models.
Better Recommendation System UI.

10. Conclusion

In our project, we crawled the review data from the IMDB website and processed the data for our recommendation system and LDA & KMeans models. We apply Natural Language Programming techniques to handle our data and save them into a dataset. The recommendation system is built based on the similarity of each movie review. After entering a movie title and the number of the results, our model and user interface can provide a movie recommendation result. We also perform a sentiment analysis and two clustering algorithms: LDA & KMeans, to make a sentiment prediction. The accuracy of both models is 0.55, but KMeans's F1-Score is slightly better. Because of this, the KMeans model seems to be more suitable for our project.

We can find out the common characteristics of popular and highly rated movies from the perspective of review analysis is mostly positive with a happy sentiment. The only negative word that shows up in the Top 20 adjective words list is "bad." Our recommendation system works smoothly; we believe that with the expansion of our dataset. We can improve our recommendation system and the accuracy of clustering models.