Movie Grading

----Based on Sentiment Analysis and Document Clustering

Kaiqi Yu, Ya Liu, Zichu Chen

Github link: https://github.com/Kelv1nYu/Moving-Grading-with-Sentiment-Analysis

1. Introduction

People may leave reviews after they watch a movie. However, there would be thousands of reviews, and it is impossible to read all. Therefore, we want to design a model to find topics in these reviews, and then predict scores in terms of each topic.

2. Background

In order to find topics, we would use k-means clustering and LDA, which are typical algorithms in document clustering. Document clustering is to assign documents to different topics. After that, we would utilize multinomial Naïve Bayes and logistic regression to predict scores for each review. This is a sentiment analysis problem. But instead of predicting the sentiment in a review as positive and negative, we predict scores with scale 0 to 5. In the end, we can get the mean score of every topic.

3. Scope

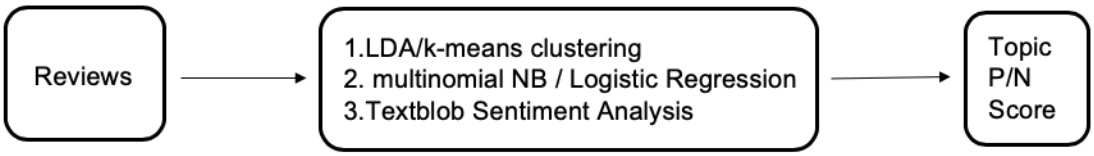

Figure 1. Scope

We cleaned our data and used several models to extract topics and analyze sentiment of these reviews. For document clustering and topic extraction, we used k-means and Latent Dirichlet Allocation model and compared the results of both of them. To classify movie reviews into positive and negative attitude, we tried Multinomial Naive Bayes classifier and Logistic regression. Also, we used Textblob Sentiment function to give a score from 0 to 5 according to each review.

4. Data Description

We used dataset Rotten Tomato reviews downloaded from Kaggle. Rotten Tomato movie review dataset has over 50,000 records and columns including id, review, rating, fresh or rotten, name of critic, top critic, publisher and the date of review. We chose only review and rating to

use and dropped other columns from the dataframe. There're 25% of rating and 10% review missing data, so we also dropped these records and still have 40,915 data record to use. The ratings in this dataset have different standards, most of which are out of 5 and out of 10 while some are given a letter as a score. Among all of these standards, rating out of 5 have 17,043 records and take the most so we chose it as our standard.

Before we put the movie into use, all the punctuations, short words and stop words were removed. We also removed those words we don't want like 'movie', 'film' because they contain no useful information. Then we tokenized the sentences and filtered all the noun and adjective for training.

5. Model Development

With data prepared, we would illustrate in detail how we develop the models. We would first do k-means clustering and LDA to get the topics, and then perform classification to predict the scores and positive / negative.

5.1 K-means Clustering

K-means clustering is an unsupervised document clustering algorithm. We first initialized k points as cluster centers, and then went through each example to choose a cluster for it based on the distance. After that, we calculated average for each group and move the cluster centers there. This process would repeat until converge.

To find the optimal number of clusters, we used grid search. Figure 2 shows that 15 would be an appropriate number. Figure 3 illustrates that how our data is clustered in terms of $k = 15$.

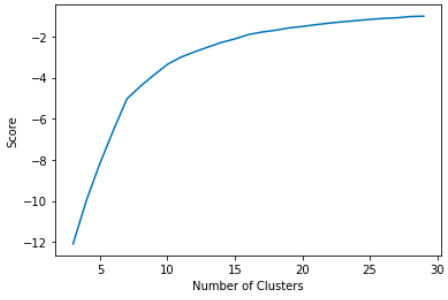

Figure 2. Hyperparameter Tuning

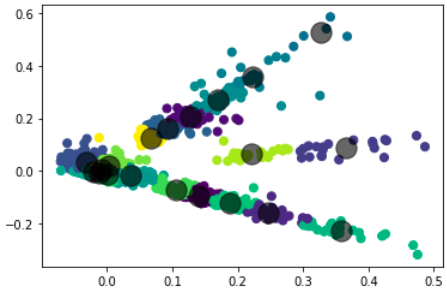

Figure 3. Visualization of Clustering

With model trained, we would like to abstract top words in each cluster so that we can get topics of each cluster. Figure 4 is a pie plot showing the frequency of each clusters. We can see that cluster 0, 11, 18, and 14 account for the majority. So, we will see these clusters in more

details. Figure 5 shows that the topic of cluster 0 might be performance. Likewise, figure 6 shows that the cluster 11 mainly focus on running time, and figure 7 illustrates that the cluster 14 contains reveiws mainly about drama.

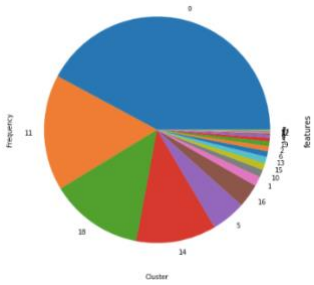

Figure 4. Frequency of Clusters

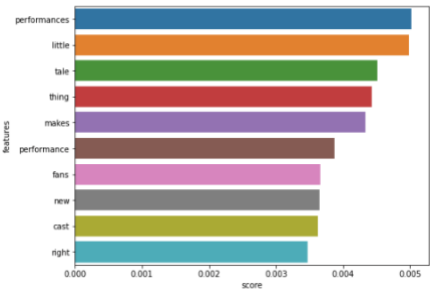

Figure 5. Top Words of Cluster 0

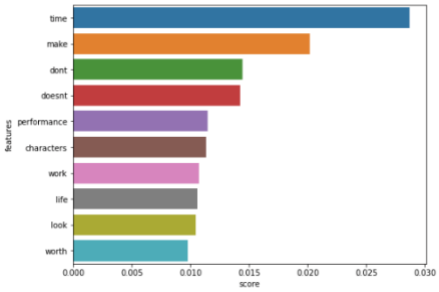

Figure 6. Top Words of Cluster 11

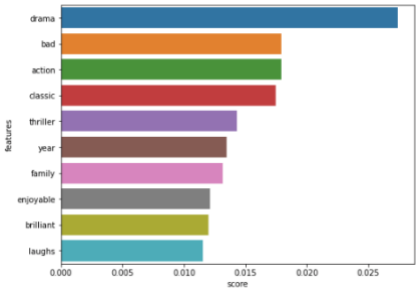

Figure 7. Top Words of Cluster 14

5.2 LDA Topic Extraction

We used LdaModel function from genism and built model for both our Rotten Tomato dataset and short paragraph.

For our dataset, we built the model and extracted topics from 17,043 records of movie review. The topics we extracted are shown below:

(0, '0.011*"classic" + 0.010*"comedy" + 0.010*"-PRON-" + 0.009*"performance" + 0.009*"sweet" + 0.007*"funny" + 0.007*"form" + 0.006*"music" + 0.006*"year" + 0.006*"top"'),

(1, '0.011*"big" + 0.011*"family" + 0.010*"funny" + 0.010*"comedy" + 0.008*"story" + 0.008*"lot" + 0.007*"drama" + 0.007*"child" + 0.007*"screen" + 0.006*"part"'),

(2, '0.015*"old" + 0.012*"character" + 0.010*"year" + 0.010*"plot" + 0.007*"game" + 0.006*"much" + 0.006*"comedy" + 0.006*"new" + 0.006*"story" + 0.006*"level"'),

(3, '0.012*"original" + 0.011*"fan" + 0.009*"sci" + 0.008*"sequel" + 0.007*"enough" + 0.007*"effect" + 0.007*"solid" + 0.007*"special" + 0.006*"star" + 0.006*"time"'),

(4, '0.017*"bad" + 0.012*"action" + 0.011*"fun" + 0.010*"movie" + 0.009*"thriller" + 0.009*"bond" + 0.008*"good" + 0.007*"book" + 0.007*"summer" + 0.006*"time"'),

(5, '0.012*"love" + 0.012*"performance" + 0.010*"story" + 0.009*"heart" + 0.008*"right" + 0.008*"drama" + 0.006*"film" + 0.006*"man" + 0.006*"war" + 0.006*"visual"'),

(6, '0.020*"full" + 0.016*"story" + 0.011*"time" + 0.010*"spanish" + 0.009*"way" + 0.009*"performance" + 0.009*"character" + 0.008*"life" + 0.008*"real" + 0.008*"actor"')

Figure 8. LDA Model result

The second topic Topic 1 has terms like family, funny, comedy, drama and children, indicating that many reviews are of family comedies. Similarly, Topic 3 indicates that some of these reviews are of sci-fi movies and talking about sequel. The result is conclusive after we removed useless information in our data preprocessing. So, we used this method to extract topic from one movie review as a function to implement in our app.

5.3 Multinomial Naïve Bayes

We tried to use the MNB model to predict the ratings of movie reviews, and found that the results were not good, so we used the model for only positive and negative predictions.

5.3.1 Text Extraction

We used two different methods for text extraction. One method is using CountVector to convert the document into a vector and the other method is using TfidfVector to convert the document through the tf-idf algorithm.

The following figure shows the shape of the converted training set and testing set:

```
[ ] cv_train_reviews.shape
(39095, 347193)

[ ] tv_train_reviews.shape
(39095, 347193)

[ ] cv_test_reviews.shape
(9774, 347193)

[ ] tv_test_reviews.shape
(9774, 347193)
```

Figure 9. The shape of the transformed training set and testing set

5.3.2 Model for Rating

First of all, we tried to use multinomial Naïve Bayes classifier to classify and predict the ratings of movie reviews, but the results were not satisfactory. As shown in the following figure, the prediction accuracy is lower than 50%.

```
print(mnb_cv_report)
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	17
1	0.40	0.02	0.05	165
2	0.51	0.53	0.52	731
3	0.34	0.23	0.27	653
4	0.48	0.76	0.59	916
5	0.26	0.03	0.05	200
accuracy			0.46	2682
macro avg	0.33	0.26	0.25	2682
weighted avg	0.43	0.46	0.42	2682

```
print(mnb_tv_report)
```

	precision	recall	f1-score	support
0	0.00	0.00	0.00	17
1	0.67	0.01	0.02	165
2	0.51	0.50	0.50	731
3	0.33	0.18	0.24	653
4	0.45	0.80	0.58	916
5	0.50	0.01	0.02	200
accuracy			0.45	2682
macro avg	0.41	0.25	0.23	2682
weighted avg	0.45	0.45	0.40	2682

Figure 10. Classification report of mnb model with data transformed by CountVector (left) and TfidfVector(right)

From the classification report, we can see that the macro and weighted average accuracy are around 0.25 and 0.45 respectively. Although the model trained on the data converted by TfidfVector performed much better than the model trained on the data converted by CountVector in predicting the accuracy of the movie reviews rated 5 and 1, both models have extremely low recall rates and F1 scores in these categories.

At the same time, both models are not ideal for predicting movie reviews with a score of 3. This shows that the model may not be easy to distinguish movie reviews with similar ratings. Therefore, we did not use this model for scoring prediction and switched to textblob for scoring.

5.3.3 Model for Positive/Negative

Then we changed the model to predict whether the film review was positive or negative, and we can see that the model's performance has an improvement.

```
print(mnb_cv_report)
```

	precision	recall	f1-score	support
Positive	0.82	0.52	0.64	3839
Negative	0.75	0.92	0.83	5935
accuracy			0.77	9774
macro avg	0.78	0.72	0.73	9774
weighted avg	0.78	0.77	0.75	9774

```
print(mnb_tv_report)
```

	precision	recall	f1-score	support
Positive	0.79	0.55	0.65	3839
Negative	0.76	0.90	0.82	5935
accuracy			0.76	9774
macro avg	0.77	0.73	0.73	9774
weighted avg	0.77	0.76	0.75	9774

Figure 11, Classification report of mnb model with data transformed by CountVector (left) and TfidfVector(right)

From the classification report, we found that the macro and weighted average accuracy are around 0.73 and 0.75 respectively. The model trained using the data converted by CountVector is slightly better than the model trained by the data converted by TfidfVector. But the gap between the two is not very big. Both models can be used in the application.

5.4 Logistic Regression

We have trained and tested the logistic regression model on the two previously converted texts, and the results are as follows.

```
print(lg_cv_report)
```

	precision	recall	f1-score	support
Positive	0.76	0.61	0.68	3839
Negative	0.78	0.88	0.82	5935
accuracy			0.77	9774
macro avg	0.77	0.74	0.75	9774
weighted avg	0.77	0.77	0.77	9774

```
print(lg_tv_report)
```

	precision	recall	f1-score	support
Positive	0.86	0.13	0.23	3839
Negative	0.64	0.99	0.77	5935
accuracy			0.65	9774
macro avg	0.75	0.56	0.50	9774
weighted avg	0.73	0.65	0.56	9774

Figure 12. Classification report of lg model with data transformed by CountVector (left) and TfidfVector(right)

From the classification report, we can see that the macro accuracy and weighted average accuracy of the model trained with the data converted by CountVector are 0.75 and 0.77, which are respectively higher than that of the model trained using the data converted by TfidfVector. Also, the second model has a high precision but a low recall and f1-score. Therefore, we chose the first model in our application.

6. Outcome

We made an application to use the implemented model. The effect of the program interface is shown in the figure. Users can arbitrarily choose a movie review as input, and then import the required model. The program will analyze the movie review and display the extracted topics, positive or negative and score of the review. (Figure 13)

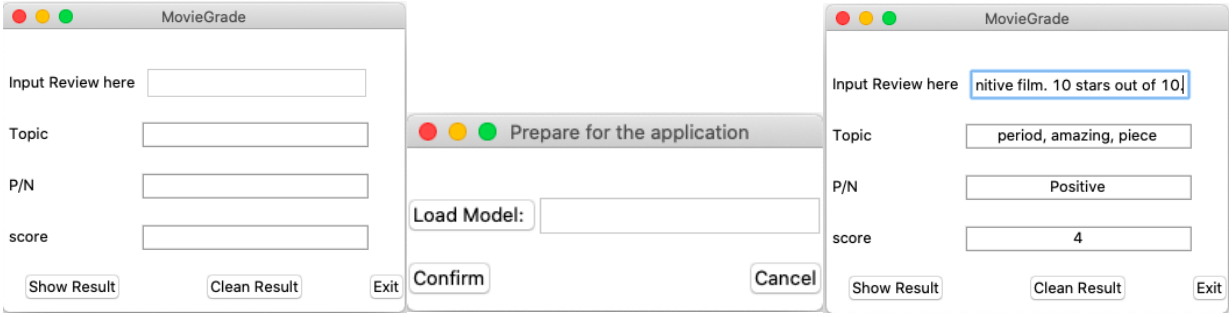

Figure 13. Outcome of Application

7. Challenges & Future Work

At first, we planned to give rating to one movie from different aspects like performance, story, character and music according to large number of reviews. However, the dataset we found is a mixture of reviews of different movies and contains no information of movie names. We also considered to use web scraping from IMDB or Rotten Tomatoes but this process was too time consuming to practice. For the future work, we want to focus on one movie at a time and combine the functions we implemented together and get scores from some aspects of this movie.

Reference

[1] Joshi, P., & Analytics Vidhya. (2019, May 6). A NLP Approach to Mining Online Reviews using Topic Modeling. Retrieved from https://www.analyticsvidhya.com/blog/2018/10/mining-online-reviews-topic-modeling-lda/

[2] Aeternae. (2019, September 2). aeternae/IMDb_Review. Retrieved from https://github.com/aeternae/IMDb_Review

[3] sklearn.metrics.precision_score¶. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_score.html

[4] Confusion matrix¶. (n.d.). Retrieved from https://scikit-learn.org/stable/auto_examples/model_selection/plot_confusion_matrix.html

[5] sklearn.metrics.plot_confusion_matrix¶. (n.d.). Retrieved from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.plot_confusion_matrix.html

[6] lakshmi25npathi. (2019, June 19). Sentiment Analysis of IMDB Movie Reviews. Retrieved from https://www.kaggle.com/lakshmi25npathi/sentiment-analysis-of-imdb-movie-reviews

[7] Devisangeetha. (2017, December 6). Analysis on IMDB movies. Retrieved from https://www.kaggle.com/devisangeetha/analysis-on-imdb-movies

[8] Kelv1nYu. (n.d.). Kelv1nYu/Project-with-Python. Retrieved from https://github.com/Kelv1nYu/Project-with-Python/blob/master/Proj2/MainPage.py