Final Project Report

Tell me how is the movie: Sentiment Analysis from IMDB Movie Reviews

Mengxi Nie, Yibu Lin

Intro to Statistical NLP_CSCI_6907_83

December 17, 2017

The George Washington University

Tell me how is the movie: Sentiment Analysis from IMDB Movie Reviews

1. Introduction

Our project is a movie recommender system which designed to do sentiment analysis using the datasets from IMDB movie reviews. According to the result of sentiment analysis, after the customer input the movie name from our movie list, our system will output whether the film is good or bad.

1.1 Background

Sentiment analysis refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine.

From the introduction, we can see the most important area of application of sentiment analysis is recommender system. A recommender system or a recommendation system (sometimes replacing "system" with a synonym such as platform or engine) is a subclass of information filtering system that seeks to predict the "rating" or "preference" that a user would give to an item. Recommender systems have become increasingly popular in recent years, and are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general. There are also recommender systems for experts, collaborators, jokes, restaurants, garments, financial services, life insurance, romantic partners (online dating), and Twitter pages. Since used in so many area, recommender system also has been used by a lot of companies such as Amazon, IMDB, Yelp and so on.

Our project is an example of movies recommender system. It collects customers' reviews from a movie review website IMDB. Using these data as training dataset to train a classifier, then judging the movie in the movie list whether it is a good film or a bad movie.

2. Project Introduction

2.1 Dataset introduction

1) The first Training set

1000 positive and 1000 negative processed reviews are collected from IMDB

2) The second Training set [1]

A dataset for binary sentiment classification containing substantially more data than previous benchmark datasets. It has a set of 25,000 highly polar movie reviews for training. In the training set, there are 12500 negative movies and 12500 positive movies.

3) Development set [1]

25,000 .txt files for testing. In the testing set, there are 12500 negative movies and 12500 positive movies.

4) Testing set

Collected from IMDB and there are 29870 .html files.

2.2 Process Introduction

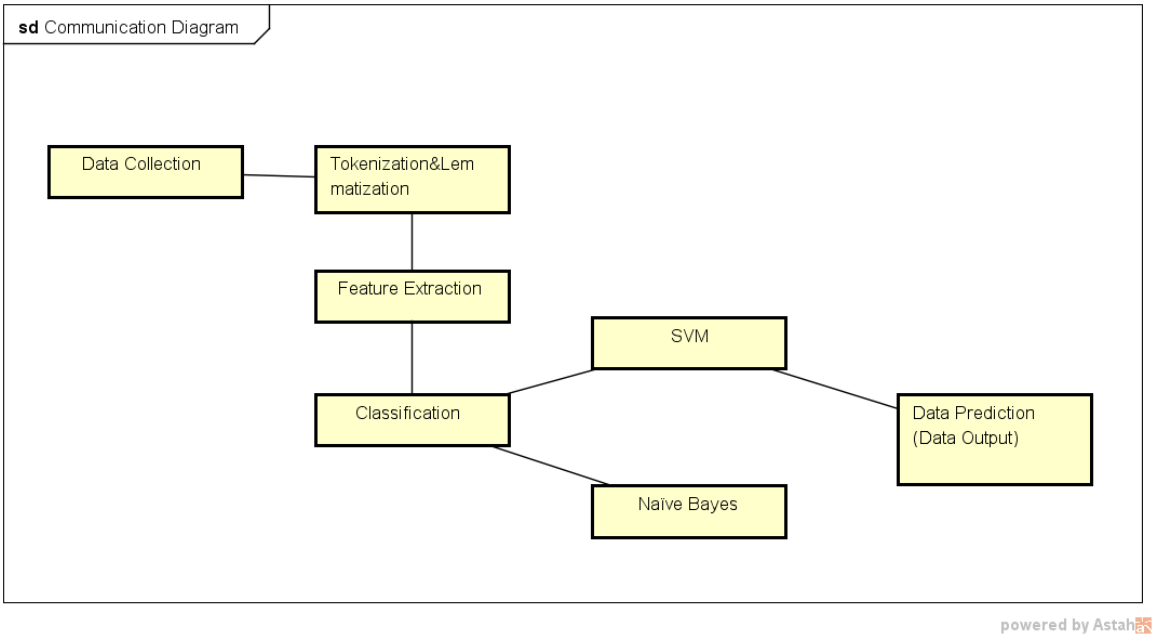

Figure 1.1 Project Process.

Our core function of sentiment analysis is divided to three parts

- Tokenization & Lemmatization
- Feature Extraction

- Classification using different classifiers
    - Naïve Bayes
    - SVM
    - GDBT
    - Data Prediction

Firstly, the system read all the files in the folder of neg and pos which collected from IMDB. Then we do the data tokenization and lemmatization. Including data filtering, stem extraction and category of word analysis. Then using countvectorizer to do word count, and using TF-IDF analysis the importance of each word in the content. Mark all the positive data with label 1 and the negative data with label 0. Next step, we used models to train the classification. we try three models: Naive Bayes, SVM and GDBT. We pick the best train models to do the data prediction. At first, we use SVM to do the data prediction. However, with the growth of training data, we choose Naïve Bayes as our classifier. Then, we output our analysis result into a .csv file.

2.3 Tokenization and Lemmatization

In the process of tokenization and lemmatization, at first, we filtered all the special symbols and the stop words. In order to increase the program operation efficiency, by the analysis of word category, the result only kept verbs, nouns, adjectives and adverbs and removes other words.

The figures show some sentences results of tokenization and lemmatization. These are same sentences. after this process, the result list decrease significantly. We think this may help our further data process and prediction.

```
['the', 'happy', 'bastard', "'s", 'quick', 'movie', 'review', 'damn', 'that', 'y2k', 'bug', '.', 'it', "'s", 'got', 'a', 'head', 'start', 'in', 'this', 'movie',
'starring', 'jamie', 'lee', 'curtis', 'and', 'another', 'baldwin', 'brother', '(', 'william', 'this', 'time', ')', 'in', 'a', 'story', 'regarding', 'a', 'crew',
'of', 'a', 'tugboat', 'that', 'comes', 'across', 'a', 'deserted', 'russian', 'tech', 'ship', 'that', 'has', 'a', 'strangeness', 'to', 'it', 'when', 'they',
'kick', 'the', 'power', 'back', 'on', '.', 'little', 'do', 'they', 'know', 'the', 'power', 'within', '.', '.', '.', 'going', 'for', 'the', 'gore', 'and', 'bringing',
'on', 'a', 'few', 'action', 'sequences', 'here', 'and', 'there', ',', 'virus', 'still', 'feels', 'very', 'empty', ',', 'like', 'a', 'movie', 'going', 'for', 'all',
'flash', 'and', 'no', 'substance', '.', 'we', 'do', "n't", 'know', 'why', 'the', 'crew', 'was', 'really', 'out', 'in', 'the', 'middle', 'of', 'nowhere', ',',
'we', 'do', "n't", 'know', 'the', 'origin', 'of', 'what', 'took', 'over', 'the', 'ship', '(', 'just', 'that', 'a', 'big', 'pink', 'flashy', 'thing', 'hit', 'the', 'mir',
')', ',', 'and', ',', 'of', 'course', ',', 'we', 'do', "n't", 'know', 'why', 'donald', 'sutherland', 'is', 'stumbling', 'around', 'drunkenly', 'throughout',
'.', 'here', ',', 'it', "'s", 'just', '``', 'hey', ',', 'let', "'s", 'chase', 'these', 'people', 'around', 'with', 'some', 'robots', '``', '.', 'the', 'acting', 'is',
'below', 'average', ',', 'even', 'from', 'the', 'likes', 'of', 'curtis', '.', 'you', "'re", 'more', 'likely', 'to', 'get', 'a', 'kick', 'out', 'of', 'her', 'work',
'in', 'halloween', 'h20', '.', 'sutherland', 'is', 'wasted', 'and', 'baldwin', ',', 'well', ',', 'he', "'s", 'acting', 'like', 'a', 'baldwin', ',', 'of', 'course',
'.', 'the', 'real', 'star', 'here', 'are', 'stan', 'winston', "'s", 'robot', 'design', ',', 'some', 'schnazzy', 'cgi', ',', 'and', 'the', 'occasional', 'good',
'gore', 'shot', ',', 'like', 'picking', 'into', 'someone', "'s", 'brain', '.', 'so', ',', 'if', 'robots', 'and', 'body', 'parts', 'really', 'turn', 'you', 'on', ',',
'here', "'s", 'your', 'movie', '.', 'otherwise', ',', 'it', "'s", 'pretty', 'much', 'a', 'sunken', 'ship', 'of', 'a', 'movie', '.']
```

Figure 2.1 Result after tokenization

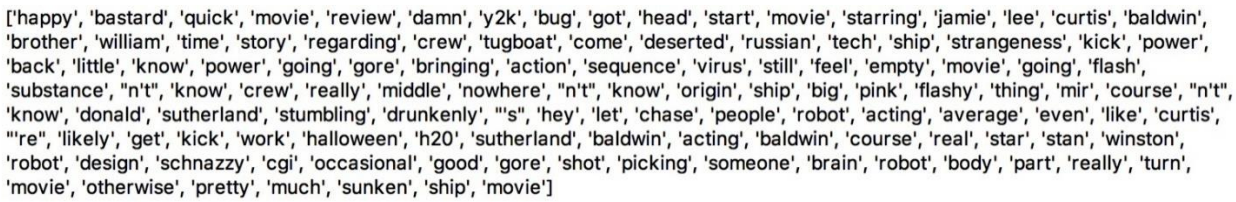

Figure 2.2 Result after lemmatization

2.3 Feature Extraction

```
# calculate the tfidf of the text
tfidf = TfidfTransformer()
x_train_tf = tfidf.fit_transform(x_train_mnb)
x_test_tf = tfidf.transform(x_test_mnb)
print(x_train_tf)
```

```
  (0, 18881)  0.0356810389742
  (0, 18757)  0.0469780915595
  (0, 18715)  0.0336331705439
  (0, 18638)  0.0637043600467
  (0, 18554)  0.0864451698582
```

Figure 2.3 Calculate result of feature extraction

We choose use TF-IDF to do feature extraction. Because there are a large number of words in a movie review, there must be much computation cost if we analyze each word. Therefore, we use TF-IDF to extract the most import words to stand for a movie review and using this result to get the feature vector of a movie review. The tf–idf is the product of two statistics, term frequency and inverse document frequency. Firstly, we construct the document-term matrix. Then, we calculate the numbers that are needed in TF-IDF. Finally, we use words with high values of TF-IDF as feature words of training data. Then, using this information to build the train model.

As the result shown in figure 2.3, we get word 0 in doc 18881 has TF-IDF value 0.03568. In doc 18554, the word 0 has a higher value and it is 0.086445.

2.4 Classification and result analysis

We have two processes to find the suitable classifier. Firstly, we have only 1000 negative movie reviews and 1000 positive movie reviews. When applying the classifier,

classifier SVM has a better result than Naïve Bayes. However, because of the small amount of training data, the precision, recall and f1-score are not good enough. The results are shown in figure 2.4.

Naïve Bayes

	precision	recall	f1-score	support
0	0.63	0.94	0.75	740
1	0.89	0.45	0.60	760
avg / total	0.76	0.69	0.68	1500

SVM

	precision	recall	f1-score	support
0	0.75	0.81	0.78	740
1	0.80	0.73	0.76	760
avg / total	0.77	0.77	0.77	1500

Figure 2.4 Classification Result

Because the scores are a little low, we use another dataset which contains 25000 reviews for data training. Additionally, we try another training model named GDBT. The new result of classification is showed as followed:

Naïve Bayes

	precision	recall	f1-score	support
0	0.86	0.83	0.85	9367
1	0.84	0.87	0.85	9383
avg / total	0.85	0.85	0.85	18750

SVM

	precision	recall	f1-score	support
0	0.89	0.70	0.79	9367
1	0.76	0.91	0.83	9383
avg / total	0.82	0.81	0.81	18750

GDBT

	precision	recall	f1-score	support
0	0.80	0.86	0.83	9367
1	0.85	0.79	0.82	9383
avg / total	0.83	0.83	0.83	18750

Figure 2.5 New Classification Result

From the new result, we can see that the result is better than before, especially the Naïve Bayes classifier. The improvement of Naïve Bayes is significant which we thought SVM should be better, but it does not. So, in the next part, we use Naïve Bayes to do the data prediction.

2.5 Data Prediction

According to the training model result, we choose the Naïve Bayes in this part to predict the data. The data of movie comments is collected from IMDB and save them into .html files. It needs to be transformed to string of text. we used the library called BeautifulSoup to support. We get the prediction result by using the *sum* of every review of a movie. That means, when this review is positive, we add 1 to the *sum* and minus 1 to *sum* when comes to a negative review of this movie. After completing all calculating, we regard a movie is positive when the sum of this movie is a positive number. Conversely, a movie is a negative one when the sum of this movie is a negative number.

```
predict_data.head()
```

	file_name	content	sentiment
0	./Desktop/sentiment/movie/0002.html	[Editor's note: Sites running 2.10 netnews wil...	positive
1	./Desktop/sentiment/movie/0003.html	Set in the offices of two psychoanalysts,...	positive
2	./Desktop/sentiment/movie/0004.html	Starring: Mickey Rourke, Robert DeNiro, Lisa B...	negative
3	./Desktop/sentiment/movie/0005.html	Harry Angel (played by Mickey Rourke) is ...	negative
4	./Desktop/sentiment/movie/0006.html	I've been a little lax in getting out mov...	positive

Figure 2.5 Parts of result of Data Prediction

The figure 2.5 shows some results of our data prediction. According to the sentiment analysis result, each movie has its own sentiment whether positive or negative and we can use these results to do the recommender function. Finally, we store the file name, content of a movie review and its sentiment into a .csv file.

3. Input and Output system

After finishing the core function sentiment analysis, we start build our input and output system to build a completed recommender system. We build a user interface and the user can input a movie and then see the result whether the movie is good or bad through the interface.

3.1 Input interface

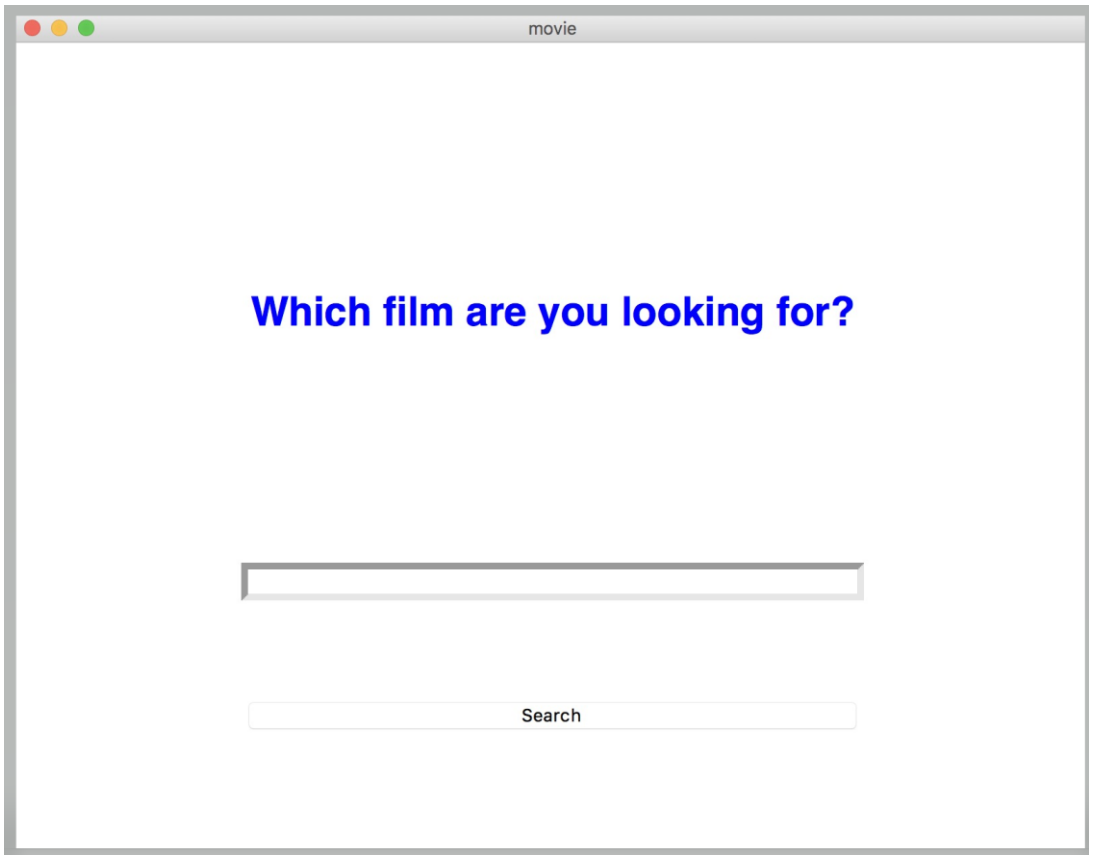

Figure 3.1 Input interface

As the figure shows, there is an input frame. We use I/O function to let customer input their movie choice.

3.2 Output results

In this part, we show some results of our output results. We test three situations, good movie, bad movie and movie that does not exist. The first one is a result of *Star Wars.* It is clearly to see that it is a good movie, and more than a half audience love this movie. Figure 3.3 shows that *Tomorrow Never Dies* is a bad movie and only a small group of audience love it. The error message shows in figure 3.4.

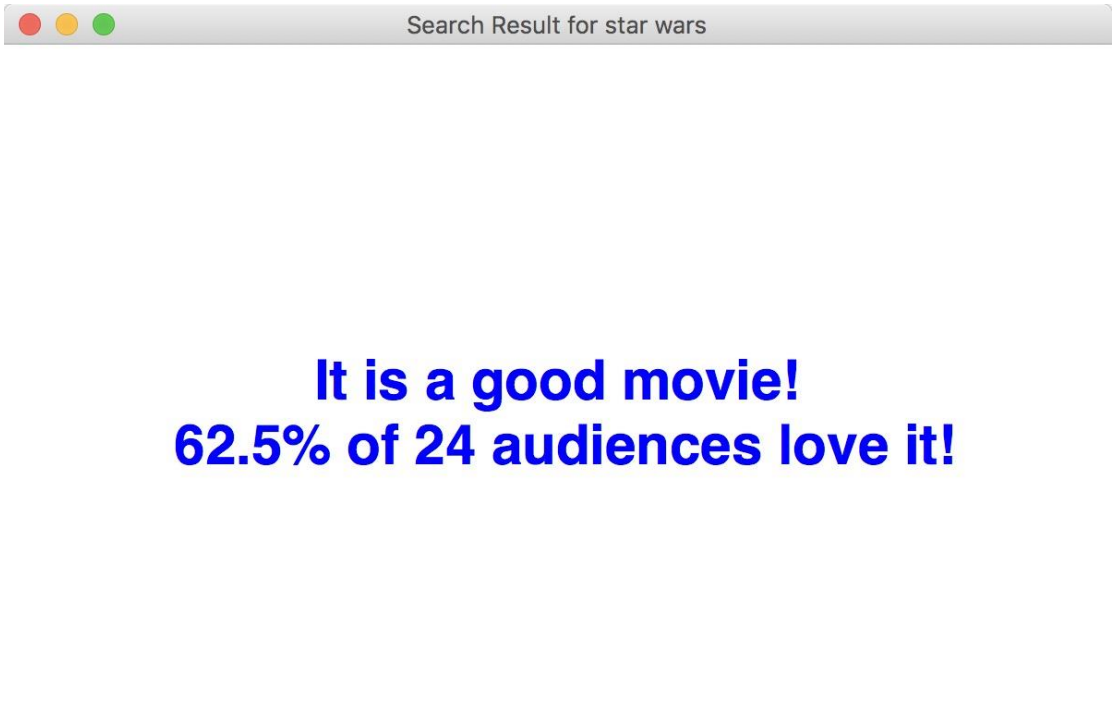

Figure 3.2 Good movie output

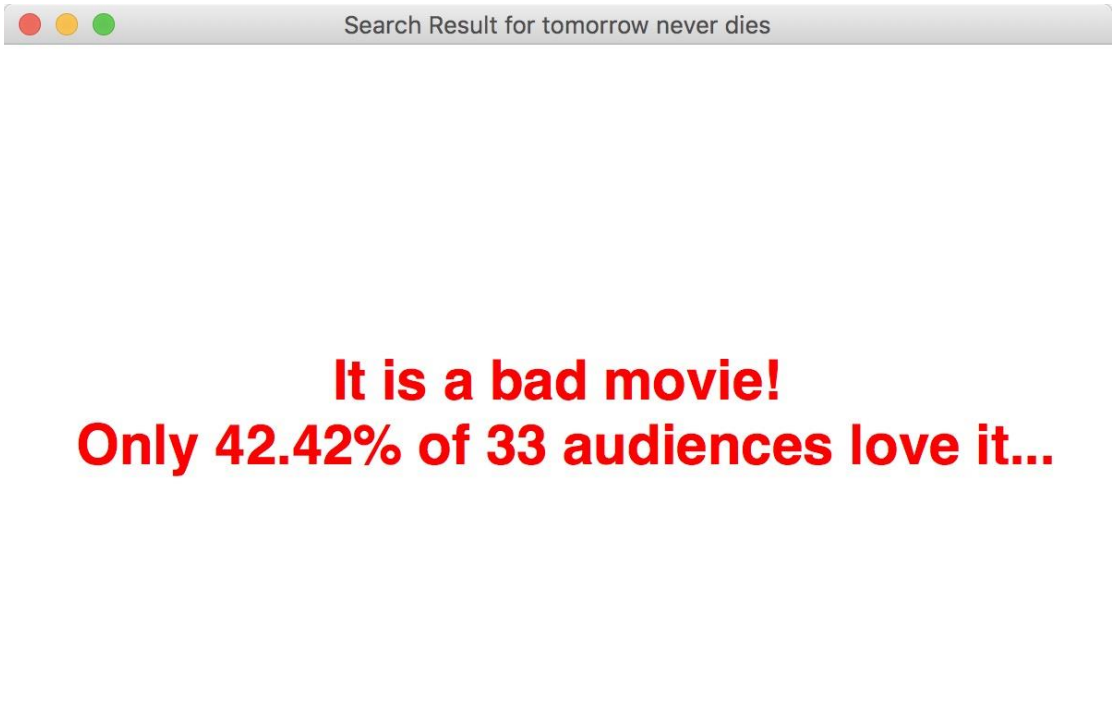

Figure 3.3 Bad movie Output

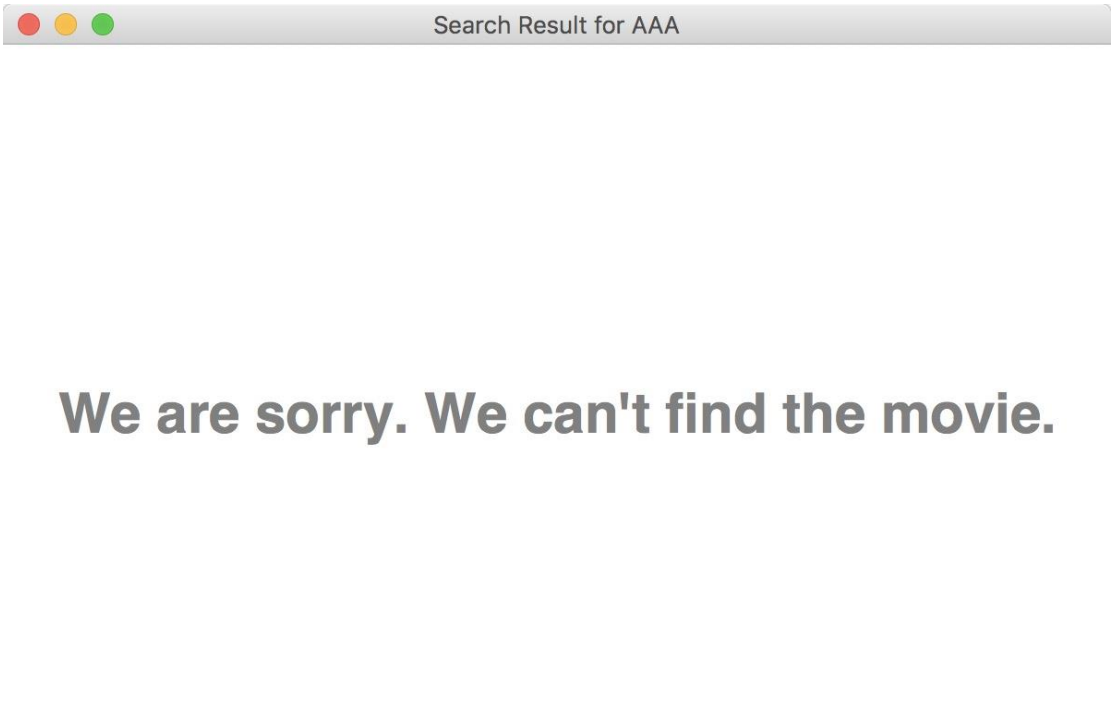

Figure 3.4 Error message output

4. Conclusion

For our project, we think our system is just a simple example of recommender system. Firstly, we collect data and do the tokenization to make analyzing easier. Then. removing some useless components to make the content of movie review more concise. Through lemmatization, we can reduce the duplication of analysis. After that, we do the feature extraction. TF-IDF is one of the important components of our system. We construct the feature vector of a movie review file using the words with high TF-IDF value. Additionally, training and testing the classifier is of great importance. Through the change of the amount of training set, we can clearly see that: 1) we should apply different classifier when comes to dealing different dataset sizes. 2) more training data will get more accurate result. We apply the best result of classifier to the reviews that need to be judged and get the final results.

5. Future work

For the future work, we have some thoughts. In this project, the movies are all before 2004 and it is a little out of date. To build a better recommender system, we may

need more reviews about the new movie as training data and we believe we get a better output results in this case. Besides, we may build a film list. Users can pick the film from out list, then get the result of the film which they choose is good or bad. Additionally, we think it should not only evolve rapidly from a very simple (positive, negative) but also to more granular and deep understanding. Such as if user can just input some emotion like" happy", the system can suggest some interesting movie which can make people happy. We think the recommender system can do a better thing in this way.

Reference

[1] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011, June). Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1* (pp. 142-150). Association for Computational Linguistics.