Feature-Based Tweets Sentiment Analysis on Movies

Chaoran Fu
Department of Electrical and Computer Engineering
Rutgers, the State University of New Jersey
chaoran.fu@rutgers.edu

Kai Kang
Department of Electrical and Computer Engineering
Rutgers, the State University of New Jersey
kk769@scarletmail.rutgers.edu

***Abstract*— Currently, the mechanism of movie review website is one score per user, which is unreasonable because the movie is made of several features such as actors, directors and music etc. Comparing to movie review websites, Twitter has much more timeliness tweets containing peoples review for each movie in theatre, thus sentiment analysis on tweets could gain many valuable results. However, feature based sentiment analysis is rarely conducted using tweets, since the 140 word limit and existence of advertisement. In this paper we propose a method that grabs tweets with different features and then conduct sentiment analysis on these tweets containing features. In this way we can implement a more comprehensive analysis on one movie. The score represents the likability of every feature. Our main contributions are: (1) We determine the feature used in the sentiment analysis; (2) We implement the sentiment analysis functions: Nave Bayes analyzer and Pattern Analyzer; (3) We move the whole system into cloud computing platform Spark using MapReduce; (4) We report the experiments on single thread and Spark and evaluate the performance of different methods.**

I. INTRODUCTION

Nowadays, with the development of movie market, millions of people would go to the movie theatre to see the newest movie. Some movie review website developed these years such as imdB and Rotten Tomatoes, offering a place to share their review for each movie. However, the mechanism of movie review website is one score per user, which is unreasonable because the movie is made of several features such as actors, directors and music. One-score mechanism is not enough to present the performance of each feature. Another issue of movie review website is the little amount of users.

Fig. 1. Home page of the movie *Creed*

From the Fig.1 we can find out that, the total amount of users rating the file Creed is only 13918, much less than the total amount of people who have seen this film. Actually, comparing movie review websites, people are willing to post their review through social networks such as Twitter. There are thousands of tweets talking about actors or other features.

Fig. 2. Example tweets of the movie *Creed*

However, the existence of advertisement would strongly affect the correctness of tweets analysis. Previously the rough sifting method can only eliminate the tweets with URLs or images. Facing this challenge, we propose the pattern analyzer to eliminate the tweets that dont have textual meaning. In the rating part, we propose the nave Bayes analyzer to rate each feature-based tweet. The training set we use is the Sentiment Polarity Dataset of Natural Language Toolkit. The result of each tweet is a probability of positive or negative. After the accumulation of analysis results for every feature we can get the final results. We use radar chart to show the score in order to show different values between features.

Besides, since the MapReduce method is more and more popular in big data analysis, We follow the trend and implement the application on the cloud-computing platform. We choose Spark as the platform because of its well support for lots of tools and libraries.

In the experiment and evaluation part, we set different sizes of dataset (varying from 20K tweets to 120K tweets) to test the performance of single thread and MapReduce. We want to determine in what data size would MapReduce beat the single thread. In addition, we design the test of MapReduce in different data partitions to find out the effeteness of them.

II. RELATED WORK

As a very popular field, lots of research on sentiment analysis has been described in the last decade.[9] The paper from Pang and Lee [1] gives a comprehensive overview of related work in that area. Actually, the topic of feature based sentiment analysis received some attention. Naveeds paper [2] identifies product features from topic keywords created

through topic classification with LDA (Latent Dirichlet Allocation) [3] and then estimate sentiment for each product feature separately.

A. Sentiment Analysis on social network

However, the product review Websites has disadvantages that those reviews are not necessarily up to date, since people dont submit their reviews immediately. Furthermore, only a few consumers review their purchases using review Web sites, so the reviews do not represent all customers opinions.

Comparing to review websites, sentiment analysis using social networking systems (Facebook, Twitter) is also increasing in popularity.[16] However, when conducting sentiment analysis on tweets or other networking services, it is not easy to grab accurate results because of the word limit, different writing styles and lack of structure of these texts. In order to overcome the shortness of social network services, Erdmanns paper [4] provides a complete solution for feature based sentiment analysis of tweets. Since their topic of analysis is evaluation of cellphones, the main problem that Erdmann has already solved is extracting features from review websites.

Another problem of sentiment analysis using Twitter is that product features are not frequently existent explicitly in tweets and are therefore difficult to extract automatically. Therefore, no feature based sentiment analysis has been used for social network, just basic sentiment analysis.

B. Cloud Computing Method

As far as we know there are not so much related work about combing the sentiment analysis and cloud computing method such as MapReduce. MapReduce is a programming framework [5] to process large-scale data in a massively parallel way. MapReduce has two major advantages [6]: the programmer doesnt know the details related to the data storage, distribution, replication, load balancing, etc.; and furthermore, it adopts the familiar concept of functional programming. The programmer must specify only two functions, a map and a reduce. The typical framework is as follows [7]: (a) the map stage passes over the input file and outputs (key, value) pairs; (b) the shuffling stage transfers the mappers output to the reducers based on the key; (c) the reduce stage processes the received pairs and outputs the final result. Due to its scalability, simplicity and the low cost to build large clouds of computers, MapReduce is a very promising tool for large-scale data analysis, something already reflected in academia.

III. SYSTEM OVERVIEW

We divide the whole progress into several parts. For each movie, we extract the specific information such as cast name and director name; next we make a list of general elements in one move, such as story, music, and visual effects. Combining two parts together we get the **Movie Features**.

Next we collect tweets from Twitter based on these features and get the **Raw Data**. We can't ignore the existence of useless tweets. Before conducting the sentiment analysis we have to preprocess the raw data. By removing the tweets that contain URL or image, we can eliminate most advertisements, thus we get the **Preprocessed Data**. At the same time we finish the data training using NLTK data set. After sifting, we do the multi-dimensional feature based sentiment analysis and get **Analysis Results**. For each movie, we calculate one overall score using different features with weights and five feature scores. Finally we use the Radar Chart to show the results in multiple dimensions.

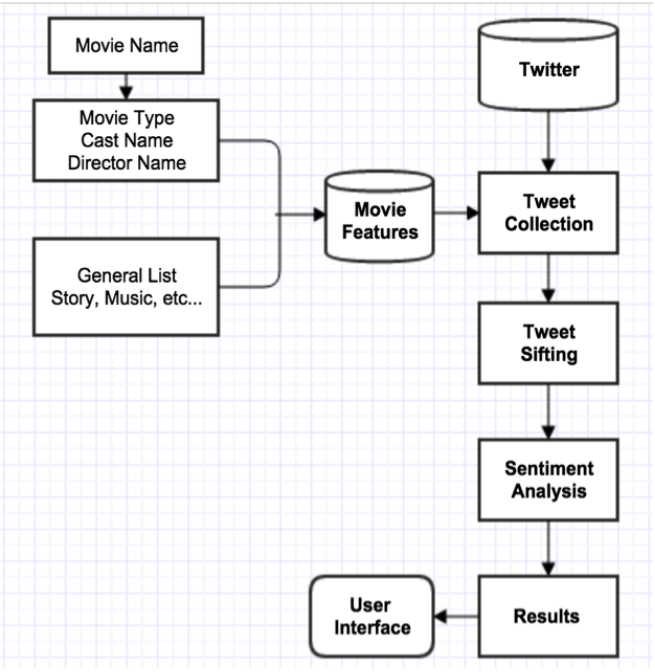

Fig. 3. The flowchart of the system

From the Figure 3 we can view the whole progress of the system. We want to point out that there is no difference for the progress between single thread and MapReduce. The only difference of them is the implementation of analyzers.

IV. ALGORITHMS

We use TextBlob [11] to perform sentiment analysis on tweets. TextBlob is a Python library for processing textual data. It provides a consistent API for diving into common natural language processing (NLP) tasks such as part-of-speech tagging, noun phrase extraction, sentiment analysis, and more. It is built on top on NLTK [12] and pattern [13] and provides easy to use APIs for developers to use NLTK and pattern functions.

A. Pattern Analyzer

In our system, we use two sentiment analysis implementations in the textblob.sentiments module. The PatternAnalyzer, which is based on pattern library, is used for analyzing the objectiveness of each tweet.

A tweet filtered out additional URL and images can be regard as a written text which can be broadly categorized

into two types: facts and opinions. Opinions carry people's sentiments, appraisals and feelings toward the world. Since the purpose of our system is to rate a movie base on peoples opinions toward a movie, tweets which carry peoples sentiments will contribute to the rate of a movie. Yet tweets which have no opinions are not necessary for sentiment analysis.

The pattern.en module bundles a lexicon of adjectives (e.g., good, bad, amazing, irritating, ...) that occur frequently in product reviews, annotated with scores for sentiment polarity and subjectivity.Polarity stands for the direction of the opinion from positive to negative. Subjectivity indicates wether a text is subjective or objective. The sentiment() function returns a (polarity, subjectivity)-tuple for the given sentence, based on the adjectives it contains,where polarity is a value between -1.0 and +1.0 and subjectivity between 0.0 and 1.0.

The sentence can be a string, Text, Sentence, Chunk or Word.In our system, the sentence is a tweet, which can be a real sentence, a phrase or several words. A tweet may even contains emoji or other abbreviations that are not real words. For our current implementation, we simply ignore emoji, abbreviation and typos which the sentiment() function can not process. The large value of subjectivity indicates strong opinions while 0 value of subjectivity indicates there is no opinion in the text and consists of purely facts in most cases. A threshold of subjectivity value should be applied to classify wether a tweet has opinion. The threshold can be lowered or raised, but overall +0.1 gives the best results for product reviews. Accuracy is about 75% for movie reviews.

```
('Tweet text: ', 'Kate Winslet in Steve Jobs')
('PatternAnalyzer result: ', Sentiment(polarity=0.0, subjectivity=0.0))
('Tweet text: ', 'Thinking Kate Winslet was the best thing about the Steve Jobs movie.')
('PatternAnalyzer result: ', Sentiment(polarity=1.0, subjectivity=0.3))
```

Fig. 4. Pattern Analyzer Results

Figure 4 shows the PatternAnalyzer output results for two actual tweets talking about Kate Winslet and Steve Jobs. The first tweet (Kate Winslet in Steve Jobs) does not contain sentiment and has a subjectivity value of 0 as expected. This tweet will be filtered out by our system. In contrast, the second tweet (Thinking Kate Winslet was the best thing about the Steve Jobs movie.) which obviously have a strong positive sentiment towards the feature Kate Winslet has a subjectivity of 0.3 and will be processed in later phases of our system.

B. Naive Bayes Analyzer

We use the NaiveBayesAnalyzer implementation for positive and negative sentiment analysis on tweets. The NaiveBayesAnalyzer is an NLTK classifier trained specifically on a movie reviews corpus. It returns its result as a namedtuple of the form: (Sentiment classification, (posV, negV)). The posV and the negV are positive value and negative value of the tweet respectively. The classification simply indicates whether the sentiment of a tweet is positive or negative.

The Naive Bayes is one of the basic approach to text classification. It is to assign to a given document d, the class $c^* = arg\ max_c P(c \mid d)$ Derived the Naive Bayes (NB) classifier by first observing that by Bayes rule is as below.

$$P(c \mid d) = \frac{P(c)P(d \mid c)}{P(d)}$$

Naive Bayes Analyzer is trained using over 2,000 movie reviews. The training was using lazy implementation so that the analyzer will not be trained until it finally has to. After training, the analyzer model generates positive and negative feature list. For every given text, or a tweet in our case, the analyzer computes the probability of each word of tweet being positive feature and negative feature. Then it integrating all the probabilities calculates the total probability of being positive and negative.

```
('Tweet text: ', 'I went to see Spectre this evening - great film, I really enjoyed it - Daniel Craig is the best
Bond ever.')
('NaiveBayes result: ', Sentiment(classification='pos', p_pos=0.6859138423914068, p_neg=0.314086157685931))
('Tweet text: ', "Saw Spectre, it was shit, Sam Smith's song is an abomination, fair play to Daniel Craig for play
ing a hot character as an ugly man.")
('NaiveBayes result: ', Sentiment(classification='neg', p_pos=0.11207117085362178, p_neg=0.8879288291463756))
```

Fig. 5. Naive Bayes Analyzer Results

Figure 5 shows NaiveBayesAnalyzer output results of two actual tweets talking about Spectre and Daniel Craig. The first tweet ("I went to see Spectre this evening - great film, I really enjoyed it - Daniel Craig is the best Bond ever.") is classified as Positive and has a positive value of 0.686. The second tweet ("Saw Spectre, it was shit, Sam Smith's song is an abomination, fair play to Daniel Craig for playing a hot character as an ugly man.") is classified as Negative and has a positive value of 0.112.

The output of this step is sentiment analysis results for each feature, containing the positive value, negative value, subjectivity and number tweets.

C. Rating Mechanism

To rate a movie, we calculate the ratio of total positive value to total sentiment value. The final score of a movie for one feature is the ratio multiply by a max score of 5.0. To be specific, we define total positive value(T_p) as the sum of positive values of tweets classified as positive tweets. The total negative value(T_n) is defined as the sum of negative value of tweets classified as negative ones. The total sentiment value is the sum of total positive value and total negative value. The rating of a movie of one specific feature is calculated using the equation below.

$$Score = \frac{T_p}{T_p + T_n}$$

Note that, using this rating mechanism, a movie does not need to have full positive value (1.0) for every tweets to achieve a max 5.0 score. A movie having no negative tweets toward it will yield a 5.0. On the other hand, if all tweets of a movie are classified as negative ones, this will lead to a 0 rating of that movie.

V. Implementation

In this section we will discuss the implementation of the system in detail.

A. Feature Extraction

The Rotten Tomatoes API provides access to Rotten Tomatoes' ratings and reviews, allowing us to build applications and widgets enriched with Rotten Tomatoes data. Using Rotten Tomatoes API, our system should be able to access the cast information of movies currently showing in theaters. There are some common features that can apply to almost all movies, such as director, story and music. For each movie, we acquire the cast features from the Rotten Tomatoes API. In addition, different movies will have different special features. For instance, movie Creed has special feature boxing. Movie Fast and Furious can have special feature car. For each movie, we will generate a feature list, contains all possible features of that movie. An example of feature list for movie Creed is shown in Table 1.

Creed	
Common Features	Director Story Music ...
Cast Features	Stallone Jordan ...
Special Features	Boxing ...

TABLE I
Feature List of Creed

Since we are planning to use radar chart for visualizing feature scores and it will not be practical to fit all features from the feature list into one radar chart, we need a mechanism to select five to six features. When the system collects tweets for one movie, rather than searching for tweets contains specific features, it will collect all tweets about that movie. When applying the sentiment analysis, the system keep tracking the count of tweets for each feature. At the data visualization phase, the system will select five most popular features ranking by the count of tweets by default. Note that our users will definitely have different interest towards different features. The system should allow users to chose specific features themselves.

However, Rotten Tomatoes is no longer issuing API keys at the time of registration. They will be reviewing each application to ensure the usage of our data aligns with their Brand Guidelines [14] and will provision keys if approved. The policy change will allow them to better service the existing developer community and to ensure that new additions are valuable partners that add to the community. We registered for a user account and applied for the access of API approximately one month before this paper is written. The Rotten Tomatoes API console approved our application several days before the project due. For our current implementation, we have not able to integrate the API into this feature extraction phase. We manually chose a new movie Creed [15] which is released on 25 November 2015 in US. We also chose five features specifically Stallone, Jordan, director, music and boxing. The movie name and features are used in later phases of our system.

B. Data Collection

We used Twitter Streaming API for data collection. The program will continuously collect tweets in real time as long as the program running as a continuous background process. We use a python program to connect to our local MySQL database and save tweets data as a txt file for sentiment analysis process.

C. Data Training

In order to get more accurate results, we choose Sentiment Polarity Dataset Version 2.0 as the training set[17]. Most of the text processing was done using the NLTK library which provided us with a useful word-tokenizer tool capable of detecting stop words and having a tool to extract the meaningful phrases from a given sentence.

There are two parts of in the training set: the Positive and Negative. Each part contains 1000 text files of movie reviews,indicating the true classification (sentiment) of the component files. Since the topic of training set is suitable of our system, we could believe that the rating accuracy would be higher than using this training set upon other topic sentiment analysis.

D. MapReduce

We implement the sentiment analysis using Spark MapReduce. The sentiment analysis contains the same repetitive work for processing each tweet. The sentiment analysis results for each tweet are independent of each other. Introducing MapReduce to our system will theoretically speed up the data processing. The overview of our MapReduce flow is illustrated by Figure **

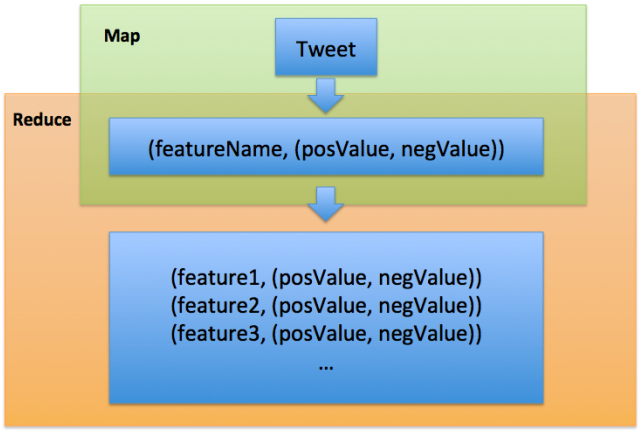

Fig. 6. MapReduce Flow Chart

Since the two analyzers need to be trained before analyze tweets, we train the two analyzer using one line dummy text before the MapReduce process begins. The map function will map each tweet to a key value pair. The key is the feature name which is contained in the tweet. As mentioned before, we are assuming each tweet only contains of at most one feature. So the map function can search for feature name in the tweet string in order. If any of the feature is in the tweet, this tweet is assigned to that feature category. If the tweet contains neither of five features chosen, the tweet will be classified as a general tweet. In this case, the generated key will be general rather than feature names. The key is corresponding to a (positive value, negative value) pair. As discussed in the rating mechanism section, the value pair will not consist of both positive value and negative value of the tweet. The value pair will have the format of either (positive value, 0) or (0, negative value). This is to ensure that only one of the positive value and negative value of the tweet will contribute to the final rate of the movie. Algorithm *.* illustrates the map function.

Algorithm 1 Sentiment Analysis

```
1:  procedure MAPREDUCE
2:  function Map(tweet t)
3:      try:
4:          analyze t
5:      except:
6:          return (error, (1, 1))
7:      if subjectivity == 0 then return (ignore, (1, 1))
8:      if feature1 in t then
9:          if positive then return (feature1, (pos value, 0))
10:     else return (feature1, (0, neg value))
11:     if feature2 in t then
12:         if positive then return (feature2, (pos value, 0))
13:     else return (feature1, (0, neg value))
14:     if feature3 in t then
15:         if positive then return (feature3, (pos value, 0))
16:     else return (feature1, (0, neg value))
17:     if positive then return (general, (pos value, 0))
18:     else return (general, (0, neg value))
```

The reducer will reduce all the key value pairs by key. It adds up all the positive value and negative value of each (key, value) pair and eventually outputs the total positive value and the total negative value for each feature. These values are used for calculating final scores for each feature as mentioned before.

E. single thread

For the evaluation of the performance of MapReduce implementation, we also conduct a single thread implementation of sentiment analysis process using python. Differ from the MapReduce implementation, this program will read the txt file as whole and process all the tweets in one sequence. The program will keep tracking and updating the total positive and negative value of each feature when process each tweet. Therefore, the single thread implementation is able to achieve one pass of tweets data. As the same as in MapReduce implementation, we train the two analyzer first using one line dummy text before processing the actual tweets data.

F. Data Visualization

We use radar chart to visualize the scores, which will provide a clear and direct expression of movie rating over different features.

VI. Experiments and Results

We conduct experiments to evaluate the performance of proposed system. All programs are running on a MacBook Air (1.3 GHz Intel Core i5, 4 GB 1600 MHz DDR3). If not specified, the MapReduce implementation is using a default partition configuration with 2 cores on local machine. The running time of all the program is calculated by setting the starting point at the beginning of the program and ending point right after all feature scores are yield. All MapReduce jobs are using cold start.

A. MapReduce vs Single thread

First, we compare the MapReduce to the single thread implementation. We test our programs using different datasets with size ranging from 20K tweets to 120K tweets. Figure *.* shows the running time of MapReduce and single thread implementation over different datasets. X axis indicates the size of dataset for each test case and the Y axis shows the running time in seconds. We collected over 20,000 tweets for movie Creed over a whole weekend and we duplicate this 20K dataset to generate synthetic datasets of size from 40K to 120K. The green line with circle dot in the Figure shows the running time of single thread implementation while the blue line with square dot stands for the MapReduce performance.

Tweets	MapReduce	Single thread
20K	37.04	20.80
40k	41.64	35.41
60k	48.99	48.35
80k	57.19	62.55
100k	65.93	77.31
120k	80.71	95.11

TABLE II

The execution time of Single thread and MapReduce

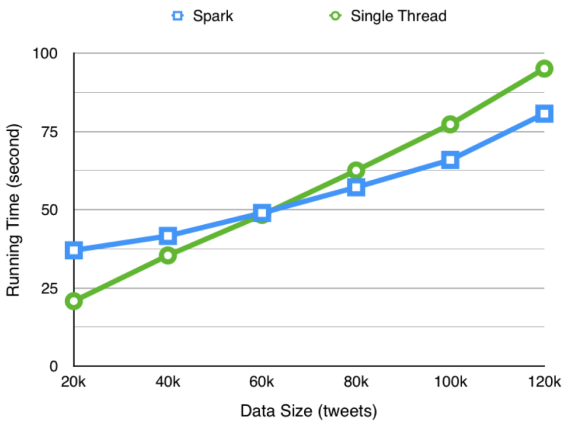

Fig. 7. Running time for MapReduce and Single Thread Implementation

Not surprisingly the single thread implementation outperforms the MapReduce one when the size of datasets are relatively small. With small dataset, the configuration and starting time of MapReduce will dominant the time complexity equation leading to a bad performance comparing to single thread implementation. As the size of dataset grow larger, the difference in performance between two programs became smaller. Two implementations have the same performance over 60K dataset. With any given dataset larger than 60K, the MapReduce outperforms the single thread implementation.

It is reasonable to use MapReduce implementation for our system in practice for two reasons. Firstly, the size of real data is significantly larger than the datasets used in the experiments. The size of dataset for one movie over one weekend is about 20K. Usually there are over 15 movies showing in theater at the same time. Collecting tweets for 15 movies over two weeks will yield over 2000K tweets. With this amount of data, it will be nearly impossible to use single thread implementation. Secondly, the MapReduce is simply using default configurations. There is a potential to improve MapReduce performance by applying optimized configurations.

B. MapReduce with different configuration

To measure the performance of MapReduce over different configurations, we conduct two sets experiments. We first run the program on 200K dataset using several different configuration settings ranging from 2 cores to 4 cores with 2 partitions to 16 partitions. The experiment results did not show significant difference in running time. All configurations have running time ranging from 105 seconds to 136 seconds. Different configuration of MapReduce will not have much impact on performance. Detailed experiment results are shown in Table and Figure.

Partitions	2-core	4-core
2	121.97	119.81
4	136.31	105.62
6	120.89	110.83
8	129.49	114.57

TABLE III
THE EXECUTION TIME OF MAPREDUCE WITH DIFFERENT CONFIGURATIONS

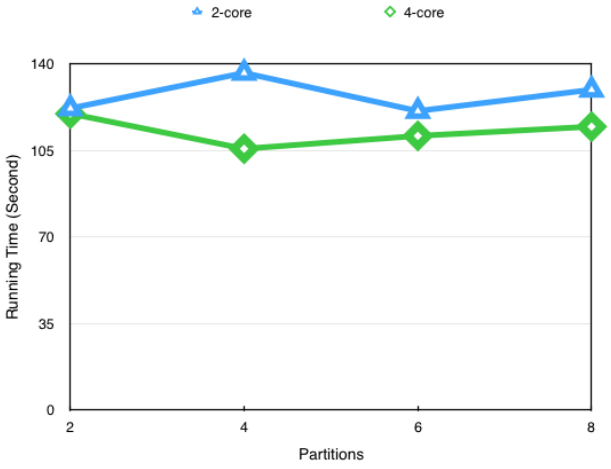

Fig. 8. Running time for different configuration of MapReduce

We also test different configuration on larger dataset. We duplicate the 20K dataset 100 times to get a 2000K dataset. This large dataset is close enough to data in practice. It is about the size of collecting tweets for 15 movies over two weeks and it is duplicated using real tweets data. The size of txt file of this 2000K dataset is 133MB. We run this dataset using 2 cores / 4 cores configuration with partition of 4. Since the default configuration for partition is 64MB for one block, a 133MB file will be partitioned into 4 parts. Experiments results yield approximate 14 minutes running time for 2 cores configuration and 11 minutes for 4 cores configuration. This shows different configurations have relatively bigger impact on performance over larger datasets.

Note that, for real system in practice, the data collection program will run on background continuously 24/7, while the updates for movie scores should be sufficient for only one time per day. Spending 20 minutes every day for updating scores is not every expensive in practice, not to mention there is still potential for improving MapReduce performance.

C. Rating Results

We collected tweets about movie Creed for four days including one weekend. The dataset have 28,752 tweets. We used default configuration MapReduce to process the dataset. It takes the program approximate 40 second to calculate the six scores including one general score and five feature scores. As mentioned earlier, the general score are calculated using tweets which do not contains any feature. The general score the movie based on current dataset is 4.0. As showed in the Figure, feature Stallone, director and music have relatively

high score while feature Jordan has a lower score. Since we calculate the scores by analyzing sentiment of tweets about this movie, the score indicates twitterers opinion toward the movie and specific features.

Feature	Score
Jordan	3.46
Stallone	4.47
Music	4.43
Boxing	4.07
Director	4.58

TABLE IV
RATING RESULTS OF EACH FEATURE

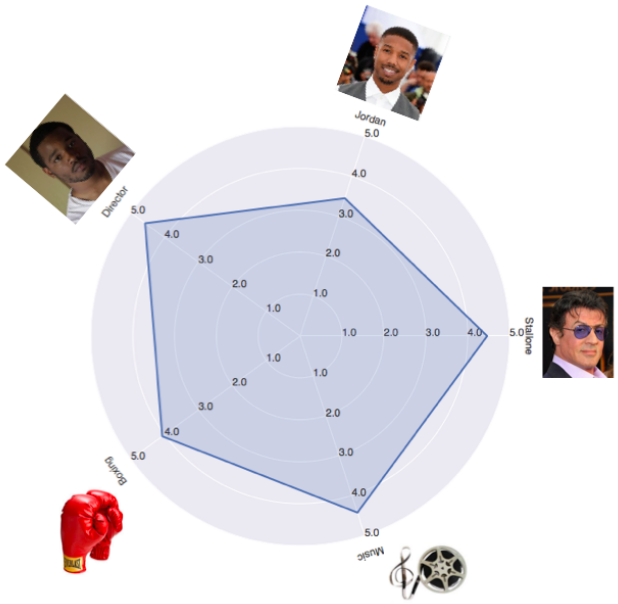

Fig. 9. Feature Based Movie Rating

From the Figure 9 we can find out the final rating results of our system. We choose Radar chart for data visualization because it can present the difference of features' score clearly.

Although comparing to the general tweets, the number of tweets containing feature is not that a lot. We still get the analysis of this movie. It can provide much more details than traditional movie review websites.

VII. CONCLUSIONS

In this paper, we introduced a method for performing feature based Tweets sentiment analysis on movies. Implementing sentiment analysis directly onto tweets is invaluable, since the existence of advertisement would strongly affect the correctness of analysis. However, because sentiment expressed on Twitter is more up to date and represents the sentiment of a larger population than movie review website such as imdB.com.

We successfully overcome the difficulty of sifting and rating tweets by Pattern Analyzer and Nave Bayes Analyzer. The method we used can greatly improve ratio of valuable tweets and the accuracy of analysis. Actually, from the analysis result of the movie Creed we can view people have different tastes and opinions obviously. Higher score of one feature means the likability of this feature.

After finishing the implementation of single thread version and MapReduce version, we find out that the trade off between them. Single thread shows the advantage under the small data set because it can save the time of configuration. However, increasing the size of data set makes MapReduce win the competition. Due to the limitation of Twitter streaming API, we could not gain more movie review tweets, but its no need to doubt that MapReduce is better solution under big data analysis.

In sum, our system shows the outstanding performance in feature based sentiment analysis of tweets. We believe the bright prospect of feature based sentiment analysis.

VIII. FUTURE WORK

In the future, we want to improve the speed of feature based sentiment analysis. The current speed is not ideal even in the MapReduce version. If we want to make our analysis a real time service we must reduce the execution time. After we evaluating the running time of each part, we hope to change the training set from large size movie review texts to corpus of sample movie review tweets with positive and negative attitude. We believe that small delicate training set would help a lot.

One problem we may encounter when we expand our system to real use as analysis multiple movies at the same time is that, it maybe too expensive to classify each tweet into different movie and features category. Since we are now classifying each tweet in the mapping function, the program has to search each feature in side each tweets. Database itself can be used for parsing tweets into different movie and feature tables. We may take the advantage of distributed system to build system with higher performance.

Apart from that, we are also interested in performing other analyzers such SVM analyzer [8]. The comparison of different analyzers would provide a comprehensive analysis.

As for cloud computing platform Spark, we could expect the better performance with more and more movies we plan to do the analysis. Also we are open to find out other good platforms, especially platforms designed for big data analysis.

In this paper we only take the tweets about movie into feature based sentiment analysis, due to the good performance of this method, we can choose other topics and get the exciting results.

ACKNOWLEDGMENT

The author would like to thank Professor Dehnavi for her instruction and support. The training set is based on the Sentiment Polarity Dataset of Natural Language Toolkit. Besides that, the author would like to thank for every developer conducting the Rotten Tomatoes API, Twitter Streaming and Textblob API.

REFERENCES

[1] Pang, B. and Lee, L., 2008. Opinion mining and sentiment analysis. Foundations and trends in information retrieval, 2(1-2), pp.1-135.

[2] Staab, N.N.T.G.S., 2013. Feature Sentiment Diversification of User Generated Reviews: The FREuD Approach.

[3] Blei, D.M., Ng, A.Y. and Jordan, M.I., 2003. Latent dirichlet allocation. the Journal of machine Learning research, 3, pp.993-1022.

[4] Erdmann, M., Ikeda, K., Ishizaki, H., Hattori, G. and Takishima, Y., 2014. Feature Based Sentiment Analysis of Tweets in Multiple Languages. In Web Information Systems EngineeringWISE 2014 (pp. 109-124). Springer International Publishing.

[5] Dean, J. and Ghemawat, S., 2008. MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), pp.107-113.

[6] Ferreira Cordeiro, R.L., Traina Junior, C., Machado Traina, A.J., Lpez, J., Kang, U. and Faloutsos, C., 2011, August. Clustering very large multi-dimensional datasets with mapreduce. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 690-698). ACM.

[7] Lmmel, R., 2008. Googles MapReduce programming modelRevisited. Science of computer programming, 70(1), pp.1-30.

[8] Koumpouri, A., Mporas, I. and Megalooikonomou, V., 2015, September. Evaluation of Four Approaches for Sentiment Analysis on Movie Reviews: The Kaggle Competition. In Proceedings of the 16th International Conference on Engineering Applications of Neural Networks (INNS) (p. 23). ACM.

[9] Blatnik, A., Jarm, K. and Meza, M., 2014. Movie sentiment analysis based on public tweets. Elektrotehniski Vestnik, 81(4), pp.160-166.

[10] Pang, B., Lee, L. and Vaithyanathan, S., 2002, July. Thumbs up?: sentiment classification using machine learning techniques. In Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10 (pp. 79-86). Association for Computational Linguistics.

[11] https://textblob.readthedocs.org/en/dev

[12] http://www.nltk.org

[13] http://www.clips.ua.ac.be/pages/pattern-en

[14] http://developer.rottentomatoes.com/docs/brand_guidelines

[15] http://www.imdb.com/title/tt3076658/?ref_=nv_sr_1

[16] Guo, H., Zhu, H., Guo, Z., Zhang, X. and Su, Z., 2010, October. OpinionIt: a text mining system for cross-lingual opinion analysis. In Proceedings of the 19th ACM international conference on Information and knowledge management (pp. 1199-1208). ACM.

[17] http://www.cs.cornell.edu/people/pabo/movie-review-data/