Large Scale Sentiment Analysis of Yelp's Rating based on Text Reviews with Spark

Yixing Sun*
y494sun@uwaterloo.ca
University of Waterloo
Waterloo, Ontario

Danning Guo
d28guo@uwaterloo.ca
University of Waterloo
Waterloo, Ontario

Bingying Xia
b6xia@uwaterloo.ca
University of Waterloo
Waterloo, Ontario

ABSTRACT

Yelp is an online crowd-sourced forum for users to rate and review businesses, which provides valuable information to users to choose where to visit or what to eat and also offer opportunities to businesses to understanding customers' needs and optimize revenue. It would be meaningful to create some machine learning models to understand Yelp free-text data using its overwhelming amount of text review from Yelp Data Challenge. In this project, we propose a sentiment prediction model based with Spark framework to deal with large scale review text data. We write a Naïve Bayes model with Laplace smoothing in Spark from scratch and also performs some text cleaning procedure in Spark. Our experiments indicate that our model achieve fast speed and scalability on large scale data compared to single processors. Test accuracy stabilizes around 92% when training set is larger than 10,000. Laplace smoothing also contributes to the accuracy.

KEYWORDS

Sentiment Classification, Spark, Yelp, Naive Bayes, Laplace Smoothing

ACM Reference Format:
Yixing Sun, Danning Guo, and Bingying Xia. 2019. Large Scale Sentiment Analysis of Yelp's Rating based on Text Reviews with Spark. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Yelp is a business directory service and crowd-sourced review forum. Businesses organize their own pages where users can rate the business from 1-5 stars and write text reviews. Users can also vote on other users useful or funny reviews and even share these reviews on Yelp and other social application.

The relationship between the reviews and the ratings is essential. Reviews contains a great amount of information about the users' opinions about the business. However, because users on Yelp may generate tens or hundreds of reviews everyday, it is impossible to invest an infinite amount of time and labor force to read through all these reviews and extract information from them. Furthermore, they are free of style, which is difficult for algorithms to understand, aggregate, and analyze.

Previous works applied several machine learning algorithms to automatically predict a Yelp review's rating on a given numerical scale based on review text alone , such as Linear Regression, Naïve Bayes, Neural Networks, SVM [7, 32, 36]. However, due to the huge amount of data, the speed of the previous studies is limited by the stand alone machine [36]. It would be significant if we could learn to predict ratings based on reviews' text alone with Spark framework to speed up the learning time without sacrificing the information. This idea could be extended to predicting movie or book ratings based on new tweets or blogs, and rating YouTube videos by mining audience comments.

2 RELATED WORK

Sentiment analysis is a process that automates mining opinions, views, and emotions from text, speeches, and other language sources through natural language processing [18]. Sentiment analysis is challenging, because it requires a deep understanding of the explicit and implicit, regular and irregular, and syntactical and semantic language rules [4]. Researchers can use sentiment analysis to determine the orientation of sentiment given text in two or more classes, where supervised learning approaches can yield excellent performance [30]. Some of the commonly applied supervised learning algorithms are Random Forest [8, 11, 12], Support Vector Machine [3, 27, 37], Neural Networks [6, 28, 33], Maximum Entropy [9, 10, 25], and Naïve Bayes [23, 34, 35].

There are several granularity levels in sentiment analysis. For document level, the whole document is classify into the respective sentiment labels. Sentence level analysis deals with tagging individual sentences with their sentiment polarities. Most recent works used the prior polarity of words and phrases for sentiment classification.

There are various applications of sentiment analysis. It can be implemented to analyze trends [2, 24], identify ideological

*Both authors contributed equally to this research.

Conference'17, July 2017, Washington, DC, USA
© 2019
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

bias [14, 20], and even target advertising and messages [19, 22]. Idea propagation through social activities is considered as an important concept [31]. Expression of opinions and reactions to ideas are influential to adoption of new ideas. Therefore, analyzing sentiment opinions on social medias, such as Yelp reviews, can help give insight to this process. Sentiment analysis and business reviews Understanding the information from the review text of Amazon products, and Yelp business and determining the intent of the author can help a company market its products as well as manage its online reputation.

Previous works using sentiment analysis and Yelp reviews dataset mainly focus on predicting stars using the text. Xu et al. [36] performed different supervised learning algorithms such as Naïve Bayes, Perceptron, and Multiclass SVM on a sample of 100,000 reviews from Yelp Challenge dataset. They implemented feature selection algorithm using Bing Liu Opinion Lexicon [21] to save computation power. Jong [15] used learning word vectors for sentiment analysis based on similarities between reviews to capture the review sentiments on a sample of 20,000 unique reviews. Fan and Khademi [7] focused on reducing the bias from different users' reviews by predicting a business' rating based on its reviews text on a sample of 35,645 text reviews. Salinca [32] built the feature extraction vector with term frequency-inverse document frequency and applied Linear Support Vector Classification, Stochastic Gradient Descent Classifier, and Logistic Regression Classifier an a sample containing 1,346,545 reviews.

Because the Yelp Challenge dataset is too large to handle on a stand alone machine, all the previous works just used a little sample from the original dataset for their studies. Consequently, previous solutions are neither sufficient not suitable for real world application. Luckily, in the past few years, large scale frameworks, such as Spark or Hadoop, has brought in immense capabilities of parallel processing in natural language processing. They are also implemented to exploit the sentiment tendency in Twitter [17, 26, 29] and Facebook [5], and display excellent performance both in accuracy and capacity. Therefore, we hope to propose a Spark framework of sentiment analysis for the Yelp review prediction problem to provide more computation power to the analysis.

3 METHOD

Idea of Naïve Bayes

Naïve Bayes models are supervised learning for probabilistic classifiers, they follow the Bayes theorem with the assumption that, the features of data are independent.

The Bayes theorem is defined as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1}$$

where A, B are some events and $P()$ is the probability function.

In order to compute the probability of A occurring given B, we need to find the probability of B occurring given A and multiply that by probability of A occurring. All of this is dividing by the probability of B occurring. Translating this to our case, we want to output the result for the sentiment given the review text. Also, due to the assumption that every pair of features is independent of each other, we transform the joint probability of features into product of the probability of each feature conditional on the sentiment. Therefore, we need the probability of the words appearing when known sentiment, the probability of each sentiment which is known as prior, and the probability of each token. Reformulate the formula into:

$$\begin{aligned} &P(sentiment|w_1, w_2, ...w_n) \\ =&\frac{P(sentiment)P(w_1, \ldots, w_n|sentiment)}{P(w_1, w_2, ..., w_n)} \\ =&\frac{P(sentiment)\prod_{i=1}^{n} P(w_i|sentiment)}{P(w_1, w_2, ..., w_n)} \end{aligned} \tag{2}$$

Actually, we don't really care about the exact value of the conditional probability, because we only want to derive the results as positive or negative. So we can ignore the denominator and use the numerator as:

$$\begin{aligned} &P(sentiment|w_1, w_2, ...w_n) \\ \propto &P(sentiment)\prod_{i=1}^{n} P(w_i|sentiment) \end{aligned} \tag{3}$$

Note that, during the sampling process, we have generate approximately same number of samples, so the prior can be ignored.

Also we pay attention to the numerical stability problem, as we need to multiply a huge number of probabilities, which would lead to increasingly small results, and this would be unfavorable because the computer might unable to process the number with high accuracy. So we use the log probability instead to take the log for both sides of the equation:

$$\begin{aligned} &\log P(sentiment|w_1, w_2, ...w_n) \\ \propto &\log P(sentiment) + \sum_{i=1}^{n} \log P(w_i|sentiment) \end{aligned} \tag{4}$$

Laplace Smoothing

With the idea of Naïve Bayes stated before, and take the log transformation of the probability, there would come into trouble when computing log(0) as we meet an unknown word. Thus, it is necessary to take the Laplace smoothing in Naïve Bayes Classification. We can see that the original

formula for the prior probability is:

$$P(w|c) = \frac{counts(w, c)}{counts(c)}$$

where $counts(w, c)$ refers to the counts of word w in class c, and $counts(c)$ refers to the counts of words in class c. The $P(w|c)$ would become problematic it would give us probability 0 when meeting the unknown words during testing process. By applying the Laplace smoothing, we define as:

$$P(w|c) = \frac{counts(w, c) + 1}{counts(c) + |V| + 1}$$

where V refers to the words in the training set. In particularly, any unknown word would have probability as:

$$\frac{1}{counts(c) + |V| + 1}$$

In general, Laplace smoothing solves the problems by adding a tiny non-zero value for the probability for both classes, and the overall sum is still 1, so the posterior would not be invalid.

Algorithm Design

This section will introduce the detailed methodology used to construct the model and test the model, and also explain the map-reduce structure. To evaluate the efficiency of the model, we also calculate the time used for each part. And the overall structure of the model is showed in Figure 1.

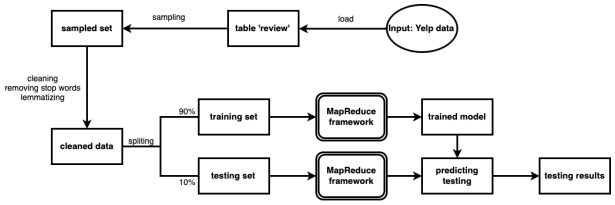

Figure 1: Algorithm flow chart

Data Load and Review. First, we load the data from Json file, the information we will use for the following computation is all contained in the table 'review', and only two columns of this table would be useful, the 'stars' and the 'text'. The range for 'stars' is from 1 to 5, and the contents for 'text' are the original comments expressed by the customers, including the punctuation, empty space, numbers, and words. Because the raw text is redundant and complicated, we need to do the following data cleaning before the training part.

Data Sampling. Due to the large size of the data set, we just need to select a subset by stratified sampling. The idea behind the sampling method is, stratifying the whole set according to its 'stars', and then returning a stratified sample without replacement based on the fraction given on each stratum. Stratified sampling is utilized because the prior distribution in the Bayes rule can be ignored.

Data Cleaning. The data cleaning process is followed by these steps:

- Cleaning: Remove empty spaces, punctuation, and numbers, covert all strings into lowercase.
- Stop words removal: Ignore some frequent words that are deemed irrelevant for sentiment analysis to speed up the models (e.g. 'the', 'a', 'and').
- Lemmatization: Reduce the inflected words properly to its canonical form (e.g. tried -> try) to improve the accuracy. And we use the function WordNetLemmatizer in the nltk module.

When the sampled data are done with the steps above, we are then ready to parse the data into the model, also we split the data with the ratio as 9:1 to training set and testing set.

Model Training. We construct a uni-gram model, each unique word in the training set is considered as a feature, and to assign the corresponding probability to each word to form the feature-probability vector for the training set. Therefore, the overall process is followed as, starting separate the training set into two parts, positive and negative group by the stars. Then count the number of tokens in the positive group and negative one, and derive the probability for each feature appearing in each group. Build the dictionary with the features and their probability derived before (e.g. bar : [('good', 0.0016415218368208463), ('bad', 0.001433129169296913)]). The size of this dictionary is the total number of the unique token. We save the dictionary as the model, which will be used for testing and forecasting later.

In our algorithm, the training result RDD is collected into a dictionary object in Python for three reasons. First one is that the probability lookup table s a read-only file and Spark will automatically broadcast the dictionary to each cluster to release the communication and computation burden. Secondly, although the probability dictionary might be large, it is bounded by the human vocabulary size. The time and space complexity of the model will be $O(1)$. Finally, the training result will be used in the testing phase. If there are two RDDs running simultaneously in Spark, it will result in block issues in partition, which might crack down the model. Therefore, we collect the probability lookup RDD into dictionary to avoid issues and reduce time.

Testing Phase. With the model derived before, and the understanding of Naïve Bayes, we use the remaining 10% samples to do the testing. And the steps for the testing phase are listed below:

- Calculate the posterior probability by mapping each token in the test set into the dictionary we created before,and sum the log value of the prior probability.

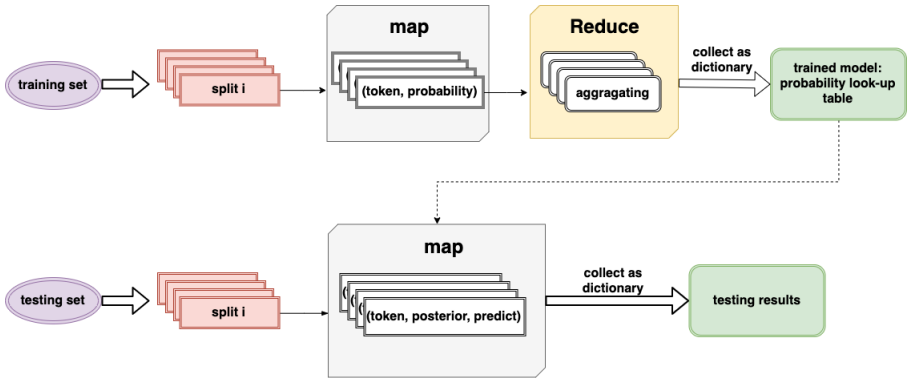

Figure 2: Spark framework

- For the tokens that can not be found in the dictionary, we apply the Laplace smoothing introduced before to avoid the break down of the programming. And for the token that only appears in one group in the trained model, then we label it with the missing one, then apply the smoothing formula to it.
- Determine the result to be positive or negative by comparing the posterior probability, if the conditional probability to be positive sentiment is bigger than it to be the negative, we assign it as the positive result.
- Return the results and calculate the accuracy compared to its original sentiment.

Spark Framework

Spark is a unified analytic engine we used here to deal with the big data challenge from Yelp. It is a powerful distributed cluster-computing framework. In our case, data are imported from JSON file into RDD in Spark. All the data cleaning, training, and testing part are done with Spark. The whole framework is shown in the Figure 2.

4 RESULT

Dataset

Our data is downloaded from Yelp Dataset Challenge [1]. This original dataset contains 192,609 businesses, 6,685,900 reviews, and 1,637,138 users. The frequency distribution of five different stars is illustrated in Table 1.

In order to perform a binary classification problem, we select all reviews with 5 stars as the positive reviews. Conversely, all reviews under 2 stars are regarded as the negative reviews. So the reviews with 3 & 4 stars are not included in the training set to improve the accuracy.

Performance Metric & Jupyter Hub Specs

We have performed the Naïve Bayes to analyze the tokens for positive and negative sentiment. We sample different size of data from the original dataset, and use 90% of the dataset for training, and 10% for testing. Because it's a binary classification problem, we only compute accuracy and record the running time. All the implementation is done on the student Jupyter Hub with AMD Opteron(tm) Processor 6380, 32 GB RAM and 64 bit Ubuntu 16.04 operating system.

Table 1: Frequency of Different Stars

Stars	Frequency	Comment
1	1,002,159	Taken as Bad reviews
2	542,394	Taken as Bad reviews
3	739,280	Ignored
4	1,468,985	Ignored
5	2,933,082	Taken as Good reviews

Experiment with 20,000 Reviews

We first generate a dataset with size of 20,000 with stratified sampling, and try to make the total number of stars 1 & 2 close to stars 5, to keep the training dataset for positive and negative set balanced.

We keep track of the time consumed in each section to evaluate how the whole process's efficiency. The results are showed in Tabel 2. We see that, using 20,000 reviews in total as the training and testing set costs a little time, and the major proportion is spent on the data loading, which is due to the extreme large size of the data and only need to done for one time. The results are quite satisfying for the speed of the processing of the training and testing process, which can give us some space on improving the accuracy by expanding the size of the dataset.

Visualization with Wordcloud

It would be more straightforward to present the words with relative high frequency to gain some general impression for the key words in different groups. Therefore, we have applied

Table 2: Computing Time for Different Stages

Stage	Time used (sec)
Data Loading	32
Data Cleaning and Sampling	22
Data Preprocessing and Training	25
Testing Phase	6

Figure 3: Wordcloud for positive tokens

Figure 4: Wordcloud for negative tokens

the "WordCloud" module to create graphs for each group. Figure 3 illustrates wordcloud for positive tokens. Words such as "love", "great", "best", "good" can be found in the plot, which imply the customers' love to the business. "service", "never", "leave", "wait" are in figure 4 as the wordcloud for negative tokens. It is also interesting to see that "love" and "like" are also in the negative wordcloud as well. However, there are negative auxiliary verbs, such as "don't" and "didn't", listed in the wordcloud to add the negative meaning to the review text.

"Each Unhappy Experience is Unhappy in its Own Way"

The number of good tokens in the 200,000 sample is 4,043,849 while the total number of bad tokens is 6,482,213. The bad tokens are over 1.5 times than the good ones. It's not an exception. We check sample with different sizes and the pattern remains the same in Table 3. It coincide with the quote from Leo Tolstoy "All happy experiences resemble each other, each unhappy experience is unhappy in its own way". Customers who are satisfied with the experience in the business are likely to just leave their good feelings and praise in the review while those who are discontented with the business tend to write down the detailed experience and anger to warn the potential future customers. Therefore, the number of bad tokens are usually greater than the good one.

Table 3: Number of Good and Bad Tokens with Different Data Size

Data size	# good tokens	# bad tokens
10,000	209,732	314,592
50,000	1,003,074	1,634,498
100,000	2,009,284	3,225,892
200,000	4,043,849	6,482,213

Accuracy

We run an experiment to see if the data size affects the test accuracy of our model. Figure 5indicates that when data size is small (<50,000 reviews), increasing data size will result in an enhancement in the test accuracy. However, if data size is larger than 50,000, the accuracy seems to be bounded by 93% and swing around 92%. It makes sense because the English vocabulary is limited, the size of the probability lookup table will eventually reach a limit with the increasing training set. Therefore, when the training set is large enough, the accuracy will be around 92%.

Smoothing

As introduced for Laplace smoothing before, we have also tested the model performance without the smoothing. We found that, in the NLTK 3.4 module in Python, it doesn't have the smoothing procedure. It treats the unknown words as none, just skips them, which might result in potential bias in the result. Therefore, an experiment is conducted to compare the results from the model stated before using Laplace smoothing in Spark with naive Bayes classifier in NLTK 3.4 without smoothing. The results are shown in Table 4.

We can see that Laplace smoothing has contributed to the improvement of the accuracy of the model for around 10% for certain dataset.

Speed Comparison with Single Processor

In order to compare the improvement of running time with Spark, we run naïve bayes algorithm with same dataset on both single processor and Spark framework to check the

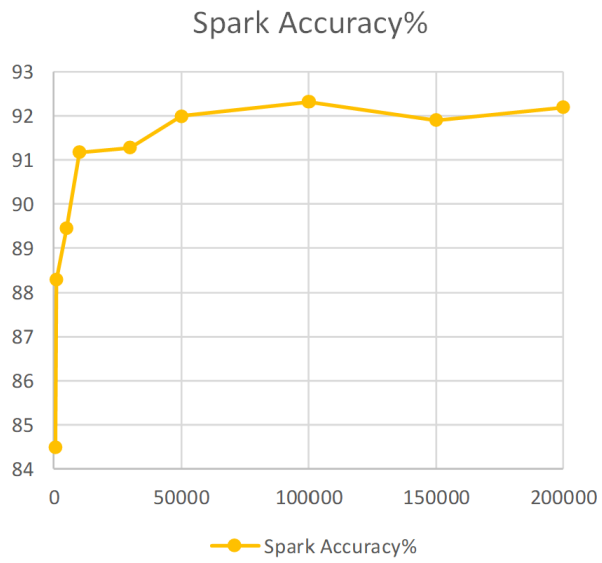

Figure 5: Test Accuracy against Sample Size

Table 4: Accuracy comparison of two methods

data size	with smoothing	without smoothing
1000	88.29%	82.78%
5000	89.44%	75.04%
10000	91.17%	80.39%
30000	91.28%	79.44%
50000	91.99%	79.29%

speed difference between single thread computing and parallelized computing. The single processor has Intel Core i5 CPU with quad core (1.6 GHz each), 8 GB RAM and 64 bit Windows 10 operating system. The naïve bayes mode made of NLTK 3.4. Due to the memory issue on the single processor, we only sampled data with till 50,000 reviews for the comparison experiment. Figure 6 illustrates that the Spark framework increases the computation speed significantly, especially when dataset is larger than 10,000 reviews. Also, the running time increases dramatically in single processor with the increasing data sets while the speed of Spark remains similar with the corresponding data sets.

One interesting point is that in small data set, single processor can defeat Spark in speed. When there are only 1,000 reviews in the data set, the speed of single processor is super fast because the data can fit in RAM. Though Spark can deal with large-scale data, it requires network communication in the cluster. Therefore, single processor outperforms Spark when it's not "big data".

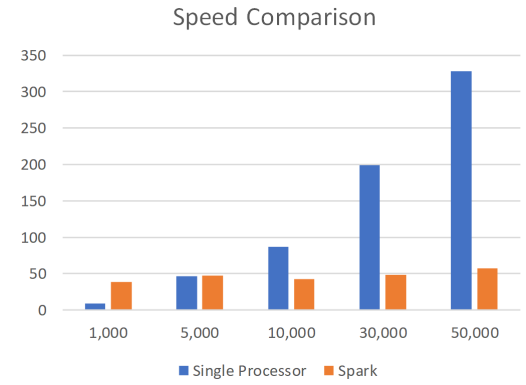

Figure 6: Speed Comparison

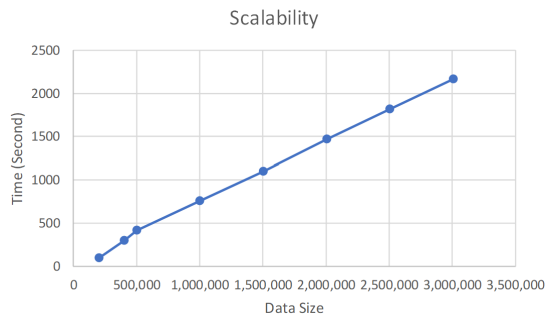

Figure 7: Scalaibility Test

Scalability

Here we investigate the scalability of our distributive naïve bayes model. We generally increase the dataset size and test the scalability only for the smoothing model. Figure 1 presents the scalability results of our model. The horizontal axis indicates the total data size used in the experiment. After the data is randomly sampled from the original file, it is split into training and testing sets with 9:1 ratio. The vertical axis shows the total time spent on the model, from pre-processing the text, training model, to finally predicting sentiment on the test set. Figure 6 indicates that our model scales almost linearly as the data size increases.

5 DISCUSSION

From the experiments above, we conclude that developers should carefully choose the data processing platform according to the data size. Spark framework definitely performs well on large-scale dataset on speed and accuracy while single processor achieve excellent result in small data. The scalability of our Spark model shows a linear performance respect to the increasing data. Test accuracy increases with the increasing training set. However, the size of English vocabulary

is limited. When data size is larger than 5,000 reviews, the accuracy rate is stabilized around 92%. Laplace smoothing redistribute the probability mass by discounting the probabilities of observed words and assign the extra probability to the unknown words. It helps enhance the accuracy of our model.

However, some limitations are still in the project. One major problem is that out model doesn't process the auxiliary negative words in reviews, such as "not", "don't", etc. This will definitely result in bias when estimating the conditional probability of each words based on different sentiment tags. Another problem is that although Laplace smoothing avoids zero probabilities, we believe that it may not be so effective as to guess an appropriate redistribution of probability mass over the set of possible words. Some studies indicate that other smoothing methods outperforms Laplace smoothing in text classification [13, 16].

6 CONCLUSION

It is important for business to understand their reviews on online crowd-sourced review forum to improve their service and optimize revenue. Yelp kindly provides their larges scale review data so that we can build computational intensive sentiment analysis models based on their data to enhance the computation speed and model accuracy. Our objective is to determine how Spark can improve the speed and how the model performs under Spark framework.

For our model, our target is to output the results for the positive or negative sentiment based on the input of reviews' text, so it's a binary classification problem. There are already many machine learning algorithms applied to this problem, and our selected method is Naïve Bayes, which is simple but efficient. We have introduced the details about the Naïve Bayes in Section 3, including the log transformation and the Laplace smoothing, both of which can improve the performance of the model and avoid the invalid situations.

For the model, it incorporates the following steps, data loading, data cleaning, training model and testing phase. We first implement the model with 20000 sampled data, and record the time costs for each step, and the results show the largest portion is spent on data loading, and the training part and testing part don't cost a lot. We also implement it with the single processor to see the time consumed. And the results show that, the Spark has a great improvement of the processing speed for larger dataset.

And for the accuracy of the model, applying the 20000 sampled data for the model can obtain the accuracy around 90%, and we see the increase in the data size can improve only a little in the accuracy. And we can conclude the model is efficient because only using 500 data points can achieve the accuracy around 85%. Comparing this model with the one without Laplace smoothing , we find the accuracy is only around 70%-80% for the same data size as the former one.

From above, our model is pretty efficient with high accuracy and high speed. Still we can improve this model with some possible actions. First, our model choose the uni-gram for the text, we only consider the separated words, sometimes it would cause confusion. So using n-gram model may improve the accuracy more, we can use pairs of tokens, three words composition or more. Second approach, from the word-cloud, we see that the negative words account a lot for the negative sentiment classification, so how can we more efficiently recognize, gather, and format the negative words in text analysis would improve the accuracy further. Third, though Laplace smoothing improves the accuracy a lot, but the large size of the vocabulary causes the probability to become very small, and we concern of the capability of dealing with the precision in Python which may cause less accurate in computing, so it may be necessary to explore other smoothing methods with higher accuracy. Lastly, our probability lookup table is read-only now, if we can create a framework with multiprocessing ability, which can avoid some updating issues, it may be more favorable in dealing with big data problem.

7 ACKNOWLEDGMENTS

We would express sincere thanks to Yelp, who generously releases their dataset for academic research. We are also very grateful to Ali Abedi, without his support we can't finish this project.

8 ONLINE RESOURCES

Our Jupyter Notebooks can be reached at https://github.com/Eason-Sun/Distributed_Sentiment_Analysis_Of_Yelp_With_Spark.

REFERENCES

[1] [n.d.]. Yelp Dataset Challenge. https://www.yelp.com/dataset. Accessed: 2019-11-10.

[2] Omaima Almatrafi, Suhem Parack, and Bravim Chavan. 2015. Application of location-based sentiment analysis using Twitter for identifying trends towards Indian general elections 2014. In *Proceedings of the 9th international conference on ubiquitous information management and communication*. ACM, 41.

[3] Abd Samad Hasan Basari, Burairah Hussin, I Gede Pramudya Ananta, and Junta Zeniarja. 2013. Opinion mining of movie review using hybrid method of support vector machine and particle swarm optimization. *Procedia Engineering* 53 (2013), 453–462.

[4] Erik Cambria, Björn Schuller, Yunqing Xia, and Catherine Havasi. 2013. New avenues in opinion mining and sentiment analysis. *IEEE Intelligent systems* 28, 2 (2013), 15–21.

[5] Sudipto Shankar Dasgupta, Swaminathan Natarajan, Kiran Kumar Kaipa, Sujay Kumar Bhattacherjee, and Arun Viswanathan. 2015. Sentiment analysis of Facebook data using Hadoop based open source technologies. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 1–3.

[6] Cicero Dos Santos and Maira Gatti. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. 69–78.

[7] Mingming Fan and Maryam Khademi. 2014. Predicting a business star in yelp from its reviews text alone. *arXiv preprint arXiv:1401.0864* (2014).

[8] Xing Fang and Justin Zhan. 2015. Sentiment analysis using product review data. *Journal of Big Data* 2, 1 (2015), 5.

[9] Geetika Gautam and Divakar Yadav. 2014. Sentiment analysis of twitter data using machine learning approaches and semantic analysis. In *2014 Seventh International Conference on Contemporary Computing (IC3)*. IEEE, 437–442.

[10] Alec Go, Richa Bhayani, and Lei Huang. 2009. Twitter sentiment classification using distant supervision. *CS224N Project Report, Stanford* 1, 12 (2009), 2009.

[11] Balakrishnan Gokulakrishnan, Pavalanathan Priyanthan, Thiruchittampalam Ragavan, Nadarajah Prasath, and AShehan Perera. 2012. Opinion mining and sentiment analysis on a twitter data stream. In *International Conference on Advances in ICT for Emerging Regions (ICTer2012)*. IEEE, 182–188.

[12] Amit Gupte, Sourabh Joshi, Pratik Gadgul, Akshay Kadam, and A Gupte. 2014. Comparative study of classification algorithms used in sentiment analysis. *International Journal of Computer Science and Information Technologies* 5, 5 (2014), 6261–6264.

[13] Feng He and Xiaoqing Ding. 2007. Improving naive bayes text classifier using smoothing methods. In *European Conference on Information Retrieval*. Springer, 703–707.

[14] Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. 2014. Political ideology detection using recursive neural networks. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1113–1122.

[15] Jason Jong. 2011. Predicting Rating with Sentiment Analysis.

[16] Alfons Juan and Hermann Ney. 2002. Reversing and Smoothing the Multinomial Naive Bayes Text Classifier.. In *PRIS*. 200–212.

[17] Amandeep Kaur, Deepesh Khaneja, Khushboo Vyas, and Ranjit Singh Saini. 2016. Sentiment Analysis on Twitter Using Apache Spark.

[18] Vishal Kharde, Prof Sonawane, et al. 2016. Sentiment analysis of twitter data: a survey of techniques. *arXiv preprint arXiv:1601.06971* (2016).

[19] Yung-Ming Li, Lienfa Lin, and Shih-Wen Chiu. 2014. Enhancing targeted advertising with social context endorsement. *International Journal of Electronic Commerce* 19, 1 (2014), 99–128.

[20] Yu-Ru Lin, James P Bagrow, and David Lazer. 2011. More voices than ever? quantifying media bias in networks. In *Fifth International AAAI Conference on Weblogs and Social Media*.

[21] Bing Liu. 2015. *Sentiment analysis: Mining opinions, sentiments, and emotions*. Cambridge University Press.

[22] Lekha R Nair, Sujala D Shetty, and Siddhant Deepak Shetty. 2017. Streaming big data analysis for real-time sentiment based targeted advertising. *International Journal of Electrical and Computer Engineering* 7, 1 (2017), 402.

[23] Vivek Narayanan, Ishan Arora, and Arjun Bhatia. 2013. Fast and accurate sentiment classification using an enhanced Naive Bayes model. In *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 194–201.

[24] Tetsuya Nasukawa and Jeonghee Yi. 2003. Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the 2nd international conference on Knowledge capture*. ACM, 70–77.

[25] MS Neethu and R Rajasree. 2013. Sentiment analysis in twitter using machine learning techniques. In *2013 Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*. IEEE, 1–5.

[26] Nikolaos Nodarakis, Spyros Sioutas, Athanasios K Tsakalidis, and Giannis Tzimas. 2016. Large Scale Sentiment Analysis on Twitter with Spark.. In *EDBT/ICDT Workshops*. 1–8.

[27] Gaurangi Patil, Varsha Galande, Vedant Kekan, and Kalpana Dange. 2014. Sentiment analysis using support vector machine. *International Journal of Innovative Research in Computer and Communication Engineering* 2, 1 (2014), 2607–2612.

[28] Soujanya Poria, Erik Cambria, and Alexander Gelbukh. 2015. Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis. In *Proceedings of the 2015 conference on empirical methods in natural language processing*. 2539–2544.

[29] Jaishree Ranganathan, Allen S Irudayaraj, and Angelina A Tzacheva. 2017. Action rules for sentiment analysis on twitter data using spark. In *2017 IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE, 51–60.

[30] Kumar Ravi and Vadlamani Ravi. 2015. A survey on opinion mining and sentiment analysis: tasks, approaches and applications. *Knowledge-Based Systems* 89 (2015), 14–46.

[31] M Rogers Everett. 1995. Diffusion of innovations. *New York* 12 (1995).

[32] Andreea Salinca. 2015. Business reviews classification using sentiment analysis. In *2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*. IEEE, 247–250.

[33] Aliaksei Severyn and Alessandro Moschitti. 2015. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 959–962.

[34] Songbo Tan, Xueqi Cheng, Yuefen Wang, and Hongbo Xu. 2009. Adapting naive bayes to domain adaptation for sentiment analysis. In *European Conference on Information Retrieval*. Springer, 337–349.

[35] Christos Troussas, Maria Virvou, Kurt Junshean Espinosa, Kevin Llaguno, and Jaime Caro. 2013. Sentiment analysis of Facebook statuses using Naive Bayes classifier for language learning. In *IISA 2013*. IEEE, 1–6.

[36] Yun Xu, Xinhui Wu, and Qinxia Wang. 2014. Sentiment Analysis of Yelp's Ratings Based on Text Reviews.

[37] Nurulhuda Zainuddin and Ali Selamat. 2014. Sentiment analysis using support vector machine. In *2014 International Conference on Computer, Communications, and Control Technology (I4CT)*. IEEE, 333–337.