Sentiment Analysis Using Naive Bayes Approach with Weighted Reviews - a case study

Brandon Joyce
Department of Computer Science
UNC Greensboro
Greensboro, NC 27412, USA
bwjoyce@uncg.edu

Jing Deng
Department of Computer Science
UNC Greensboro
Greensboro, NC 27412, USA
jing.deng@uncg.edu

Abstract—**Online reviews are critical in many aspects, for business as well as customers. Yet the accuracy and trustworthiness of these reviews are usually unsubstantiated and little research has been performed to investigate them. In this work, we use a set of Yelp reviews on various topics (food, hotel, etc.) as an example to perform sentiment analysis and investigate the correlation between review comment sentiment and its numeric rating. We use feature selection techniques to statistically remove redundant words from reviews, thus improving run time and accuracy. Our method gives higher weight to those terms/words appearing in reviews with more useful votes. These techniques combined with Naive Bayes approach achieves an overall accuracy of 75%. More interestingly, our method is shown to perform well in 1-star and 5-star reviews, with a 92% accuracy for the latter. With such a strong accuracy, we argue that the proposed sentiment analysis technique can be used to shed light on all online comments, especially those without numerical ratings.**

Index Terms—**Natural Language Processing, Naive Bayes Classifier, Sentiment Analysis, Social Media, Yelp**

I. Introduction

Online social networks play increasingly important roles in our daily lives. We choose restaurants and lodging based on user reviews and we also try to provide helpful reviews for others to decide. However, it is unclear how such reviews' sentiments correlate with the review ratings accompanying them. For instance, are all 5-star reviews the same? Should people disregard all 3-star reviews? When facing several detailed reviews, which one or ones should be trusted more?

These were the questions that motivated us to investigate user reviews and we focused on Yelp reviews due to a number of reasons. Yelp is a popular social media site where users can post reviews about companies, such as restaurants and hotels. Other Yelp users can respond to these reviews, for example, by indicating if the review was helpful or funny.

The goal of this research is to predict the positive or negative sentiment of Yelp reviews. Our approach is to use a Naive Bayes classifier. We use feature selection techniques to statistically remove redundant words from reviews, thus improving run time and accuracy. We also weight reviews with more useful votes higher than those with fewer votes.

The paper is organized as follows: Section II discusses state-of-the-art works in the field. We explain our methodology in Section III. Our tests and results are presented in Section IV and concluding words are provided in Section V.

II. Related Work

There have been some work in different review sentiment analysis, such as aspect identification in reviews [1], frequently co-occurring entropy [2], opinion mining [3], word vector analysis [4], dimension impacts [5], and big data analytics [6]. More specific works are discussed below:

In [7], Dey et al. used 10,000 movie reviews and 10,000 hotel reviews in two separate experiments. Each dataset had an equal distribution of positive and negative reviews. A Naive Bayes Algorithm was used with a top performance accuracy of about 82% for the movie reviews and 55% for the hotel reviews.

In [8], Bakhshi et al. examined the social evaluation of Yelp Reviews. A Yelp user can interact with a review and vote for the review being Useful, Funny, or Cool. They found that active and regular members were the highest contributors to high-quality reviews. A strong relationship with the number of reviews that a user wrote and the number of useful votes was identified, suggesting that more experienced users write more reviews that are deemed helpful.

In addition, Lei et al. [9] relied on users' social circle/network to calculate the interpersonal sentimental influence of their friends in order to make better predictions. Thus, a user's preference is considered closely correlated with those of their social network friends.

In *Feature Selection*, Naive Bayes Classifier assumes that each feature/word is independent. In [10], Uysal first performed global feature selection by implementing either Information Gain, Gini index, and Distinguishing feature selector on their text dataset. Afterwards, they would perform a local feature selection process. This was done by partitioning the data by class, and using an odds ratio metric to determine the top features in each class. These two feature sets were combined and Naive Bayes or Support Vector Machine (SVM) would be used for text classification.

In [11], Salinca was able to achieve an accuracy around 90% using a Naive Bayes classifier. Further improvement was achieved with negating words that were preceded/succeeded by a negation word, e.g. "not". In her study, three-star reviews were dropped in order to improve accuracy.

Delta TFIDF [12] is a scheme close to our design. In Delta TFIDF, term frequency transformation was used to boost word importance that are unevenly distributed between positive and negative categories. It was implemented in Support Vector Machine (SVM) and evaluated in several balanced datasets.

Compared to these works, our approach differs from the following perspectives. We apply a *linear* feature selection process (no exponents, logarithms, or division of feature frequencies). For example, information gain applies a logarithmic function. We instead argue that by not applying a logarithmic function, we can take advantage of the exaggerated differences of word frequencies. In essence, a word with a higher frequency appears in more reviews, so even if a less frequent word contains slightly more information, we would not be able to use it in that many reviews. Furthermore, we take advantage of the contextual information in Yelp reviews, such as the number of user votes for a review. We further treat different Yelp categories as local groups. Reviews for specific categories contain local words that are unique for themselves and may contain unique sentiment. When they are treated differently within the local groups, more accurate results can be achieved.

III. Sentiment Analysis Methodology

A. Data Preparation

In this work, we used the Yelp challenge dataset, round 13 [13]. First we need to decide what review ratings should be considered as "negative" and "positive". There are altogether 5 simple different ratings, 1-5. Since most likely costumers leaving a rating of 3 is unlikely to have liked or enjoyed the service, we map all reviews with ratings of 1-3 as "negative" and 4-5 as "positive".

For our experiments, a 70/30 data split was used for training/testing purposes. There are 6,685,900 reviews in the Yelp challenge dataset, only a small number of which were removed due to bad formatting during the importing process. Then we removed any graphical charters and converted the text encoding to UTF-8. We replaced all special characters with a space, except for “space" and “apostrophe." We left spaces alone, and replaced “apostrophes" with the empty string. This was done in order to preserve negative words such as “don't" and “won't." All review texts were then converted to lowercase.

We used the R tm package [14] to remove stopwords (“and," “or" etc.), and we also performed stemming, so words like “love," “loving" and “loved" would all be represented by the same word “love."

We used reviews from the top 1,000 businesses, for a total of 1,127,333 reviews. As a perspective, in order to make the numbers of positive/negative labels equal, we would have 681,414 reviews (there were more positive reviews overall).

B. Naive Bayes Algorithm

Naive Bayes Algorithm is based on the following formula:

$$P(\text{label}|\text{feature}) = \frac{P(\text{label})P(\text{feature}|\text{label})}{P(\text{feature})}$$

Naive Bayes algorithm is used as the baseline in this work. There are two labels in the reviews: positive and negative. The features are the words in the Yelp reviews. Note that we removed stopwords from Yelp reviews. Naive Bayes Algorithm makes the “naive” assumption that any given word/feature has an independent probability from another word/feature. Thus, the $P(\text{label}|\text{feature})$ for a Yelp review containing n words can be calculated as:

$$P(\text{label}|\text{review}) = \frac{P(\text{label})P(t_1|\text{label})...P(t_n|\text{label})}{P(\text{review})}$$

We used the National Language Toolkit (NLTK) to implement Naive Bayes Algorithm [15].

C. Negating words

Since Naive Bayes Algorithm assumes that features are independent, it cannot easily distinguish the word “good" in the contexts of “that was very good" and “that was not good." We look at several words preceded by “negation words" (not, no, didn't, won't, etc.) and negate them, so in the previous example “that was not good" would be converted to “that was not NOTgood." This is similar to [11], but we do not (explicitly) negate any words before a negation word, rather directly after a negation word. We also applied this negation as the last step of our feature selection process. We converted words to lowercase, so the uppercase negation would not be found naturally in the dataset. Furthermore, we kept the original negation word to aid in the review being classified as negative, while at the same time the negated word should help prevent the review from being labeled incorrectly, i.e., as positive.

D. Information Gain

We used the information gain formula presented in [10]. The information gain (IG) of a word/term t is defined as the following:

$$\begin{aligned} IG(t) \quad = \quad & P(t)\sum_{i=1}^{M}(P(C_i|t)\log P(C_i|t)) + \\ & P(\bar{t})\sum_{i=1}^{M}(P(C_i|\bar{t})\log P(C_i|\bar{t})) \end{aligned} \tag{1}$$

where M is the number of classes (2 in our analysis), C_i represents a particular class (just positive and negative in our study), and $\bar{t}$ represents the event of absence for t.

To calculate information gain, we separated the data into two sets: positive labels and negative labels. We then converted each review into a unique array/vector of words. Then we used R's table function [16] to count the number of times a word appeared in the positive set and the negative set. Using these frequencies, we were able to calculate information gain, IG.

E. Simplifying Information Gain for a two-class dataset

The Information Gain formula as in (1) could be further simplified by just looking at the difference of the number of times a word appears in a positive review and that in a negative review. Thus, if a word appeared equally in each class, the difference would be zero and we would know this word possessed no relevant information that could help us distinguish reviews containing them between classes. Likewise, if a word only appeared in positive reviews, then it would have a higher score. However, a word that only appeared 50 times in positive reviews would be weighted less than a word that appeared 1,000 times in positive reviews and only 50 times in negative reviews. We call this difference “word sentiment magnitude", termed η. The difference of this with Delta TFIDF [12] is that we do not use division instead of subtraction, apply a logarithmic function to our quotient/difference of positive and negative words, and we do not scale our results by the total frequency of the word. Furthermore, the Delta TFIDF results presented in [12] assumed balanced datasets, which can be unrealistic to require. We can also scale the positive or negative class with η, so we do not have to assume that each class contains equally informative word frequencies.

By calculating word sentiment magnitude, we could rank words more intelligently than just by ranking words by their frequencies. Also, unlike the information gain metric used in [10], we could calculate a list of positive and negative words to form a better distribution of words than information gain that might favor one class of words.

For a given term/word t in a 2-class dataset with an equal number of classes, its word sentiment magnitude is calculated with the following formula:

$$\eta(t) = f^+(t) - \eta_0 f^-(t) \tag{2}$$

where $f^+(t)$ and $f^-(t)$ are the numbers of times term/word t appears in positive and negative labeled Yelp reviews, respectively, and η_0 is a constant that can be adjusted for different weights between $f^+(t)$ and $f^-(t)$.

We also extended this process to work with uneven classes by scaling the frequency of words that appeared in the smaller class. Assume that all reviews are classified into two different sets, positive set $\mathcal{R}^+$ and negative set $\mathcal{R}^-$, with L^+ as the total number of positive review, $L^+ = \|\mathcal{R}^+\|$, and L^- as the total number of negative reviews, $L^- = \|\mathcal{R}^-\|$, then we could calculate word sentiment magnitude with the following formula:[1]

$$\eta(t) = f^+(t) - \frac{L^+}{L^-}\eta_0 f^-(t) \tag{3}$$

We could calculate the word with the most “positive" impact by finding those words with the highest word sentiment magnitude. Similarly, the most “negative" impact words are those with the most negative word sentiment magnitude.

F. Weighting Reviews

Furthermore, we could take advantage of useful votes, as in [8] and [9] to support our use of weighting reviews. In [8], their research shows that useful reviews may have higher quality, since they are probably written by active users (as opposed to a non-active, bot, or fake user). In addition, we expect that users on Yelp might have some friends interacting on reviews, in addition to other means of social interaction, forming a type of social network. In essence, we are using an approach that is reverse of that in [9], in which Lei et al. used the sentiment of friends to make a prediction/suggestion. We take "useful votes" (someone who found the review useful) as an indicator of sentiment for that friend/follower. So, if someone found the review useful, they might share a similar sentiment for that business, and we can weight this review higher (as if several people wrote the same review). Thus, instead of constructing a social friend network (which may be computational expensive), we simply use the useful votes as indicators for that hidden/uncalculated social network.

It follows that we can weight each review by increasing the number of times in appears in the sample. We choose the Yelp review feature “useful votes” as the weight (an integer representing how many people voted for the review as useful). Again, the motivation behind this is that people who found the review useful can “relate” to the review (i.e. they wrote a similar review, had a similar experience, or they believed the sentiment expressed in the review matched the star rating). This should help mitigate fake or irrelevant reviews by increasing the more relevant words that appear in highly useful reviews.

Given a review r with $u(r)$ useful votes, we duplicate this data point $\log u(r)$ times. In essence, we are increasing the probability of words that appear in useful reviews. Our assumption is that these words are more meaningful than words that appear in a review with fewer or even without “useful" votes. Reviews with more “useful" votes should contain words that accurately describe the sentiment of the view (positive or negative). We choose a log function to prevent reviews with extremely high useful scores from dominating the training dataset, which could lead to over-fitting.

Given a set $\mathcal{R}$ of n reviews that can be partitioned into a set of positive reviews $\mathcal{R}^+$ and negative reviews $\mathcal{R}^-$. We will also use an indicator variables t_r which will be 1 if a term t appears in review r and zero otherwise. Combining weighting reviews with (3), the weighted Word Sentiment Magnitude becomes:

$$\begin{aligned}\eta(t) &= \sum_{r\in\mathcal{R}^+}[(1+\log(1+u(r))t_r] \\ &- \frac{L^+}{L^-}\eta_0 \sum_{r\in\mathcal{R}^-}[(1+\log(1+u(r))t_r] \end{aligned} \tag{4}$$

G. Tested Techniques

In this work, we focus on experimenting different combinations of the above techniques. These are discussed below:

First of all, Negating Words are performed through preprocessing. This is important such that word sentiments will not be mis-classified. Word Sentiment Magnitude (WSM) is then investigated because of its potential. This technique takes care of words showing up in highly positive or highly negative reviews and such will be counted. Finally, Useful Votes are used in addition to evaluating ratings on reviews and how these ratings are reflected on review content quality. Combining

[1]We would ignore the trivial case of $L^- = 0$.

these techniques, the new sentiment analysis technique is called Weighted Word Sentiment Magnitude (wWSM).

The baseline technique is Naive Bayes with little feature selection. We did remove words that occurred in fewer than 50 reviews (we went from 38,848 total words to 11,059).

IV. Performance Evaluation

First, we present the list of words with the highest and lowest Word Sentiment Magnitude, η, as defined in (3), with $\eta_0 = 1$. Table I lists all top-10 positive and top-10 negative words that we identified in the Yelp dataset. While the list of these positive words looks normal, the negative word list does have some surprises: the word "just" is the top negative word and even some of the positive or neutral words are on the list, e.g., "get", "like", and "better."

TABLE I: Word Sentiment Magnitude (with $\eta_0 = 1$): Top 10 Positive and Top 10 Negative (unstemmed for clarity)

positive word	η	**negative word**	η
great	44,194	just	-31,083
delicious	33,956	didnt	-28,984
love	33,290	like	-27,471
amazing	32,906	ask	-26,379
best	25,239	bad	-23,685
perfect	22,194	dont	-23,589
definite	21,372	get	-23,320
favorite	18,179	better	-22,929
awesome	16,899	us	-22,329
recommend	15,235	said	-21,623

Next, the list of popular negation words is shown in Table II. When the negation technique is used, any word preceded by any of these words would have a reversed sentiment (positive to negative, or negative to positive).

TABLE II: List of Negation Words

wont	wasnt	wouldnt	werent	no
not	never	doesnt	cant	cannot

Information Gain has been used in some of the techniques and we only kept words with the highest information gain score (top 1,000 words). Note that such a score does not distinguish between positive/negative words. We show the top 20 words with the highest information gain value in Table III.

In Table IV, we summarized the average number of useful votes for reviews with different star ratings. It is interesting to see reviews that gave out lower ratings such as 1-star or 2-star received more useful votes. Five-star reviews received the fewest useful votes. This could have been caused by the fact that, when people submitted strong reviews, they simply wrote a few praising sentences. While those reviewers giving out lower star ratings might have felt that more explanations were warranted and thus provided more details, helping them to receive more useful votes. Therefore, there exists a “default is excellent” sentiment. Overall, the average of useful votes for all reviews is about 1.26. Also included on Table IV are the number of reviews in different star ratings.

TABLE III: Information Gain Words: Top 20

word	**IG**	**word**	**IG**	**word**	**IG**
delicious	0.6035	worst	0.5966	said	0.5947
amazing	0.6023	ask	0.5961	terrible	0.5938
great	0.6009	bad	0.5961	minute	0.5936
love	0.5984	didnt	0.5960	favorite	0.5935
ok	0.5974	nothing	0.5956	disappoint	0.5933
perfect	0.5967	horrible	0.5954	best	0.5932
told	0.5966	rude	0.5947		

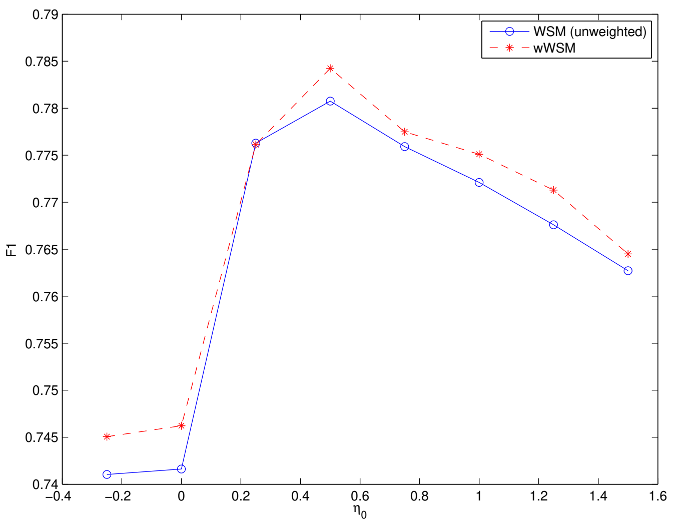

Fig. 1: Comparison of different η_0 on unweighted WSM and weighted WSM (wWSM). Note the small range of y-axis for the purpose of clarity.

TABLE IV: Average useful votes and total reviews with each star rating

	1-Star	2-Star	3-Star	4-Star	5-Star	All
mean	1.67	1.41	1.28	1.27	0.97	1.26
#rev.	73,362	64,429	100,569	90,341	148,019	476,720

Next, we investigated the impact of different η_0 values. We tried $\eta_0 = -0.25$ to 1.5 with 0.25 gap for two of the techniques that we developed, unweighted WSM and weighted WSM (wWSM). The results are shown in Figure 1 in F1 scores. The F1 scores of both schemes improve as η_0 increases, until $\eta_0 = 0.5$, at which point the F1 scores peak then decrease with further increase of η_0. Such a change could have been caused by the impact of words showing up in negative reviews with marginal meanings. We will focus on $\eta_0 = 0.5$ for the rest of our investigations.

Similarly, Figure 2 shows the accuracy results for reviews with different star ratings when different η_0 values are used. Except for 3-star reviews, all reviews are predicted with better accuracy when $\eta_0 = 0.5$.

Table V presents the comparison between Delta TFIDF and the weighted WSM scheme that we designed. FP, FN, TP, and TN are false positive, false negative, true positive, and true negative, respectively. Working on balanced datasets, the weighted WSM scheme does not show much improvement over delta TFIDF scheme. However, in unbalanced datasets, the weighted WSM scheme clearly outperforms the Delta

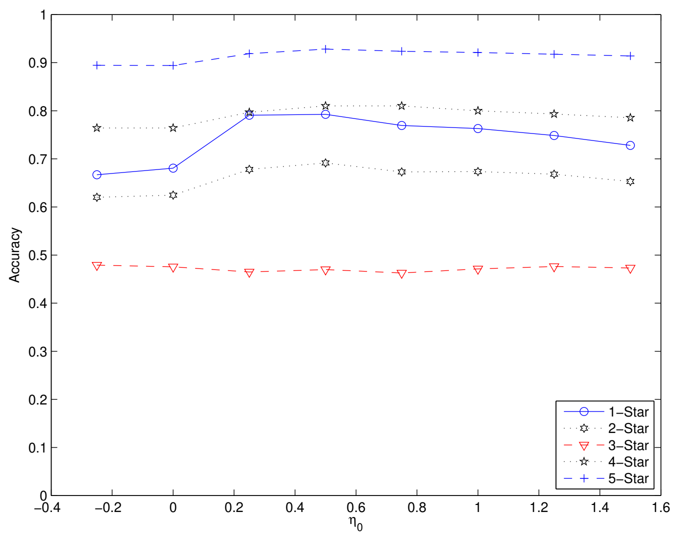

Fig. 2: Accuracy for reviews with different star ratings in the wWSM scheme with different η_0.

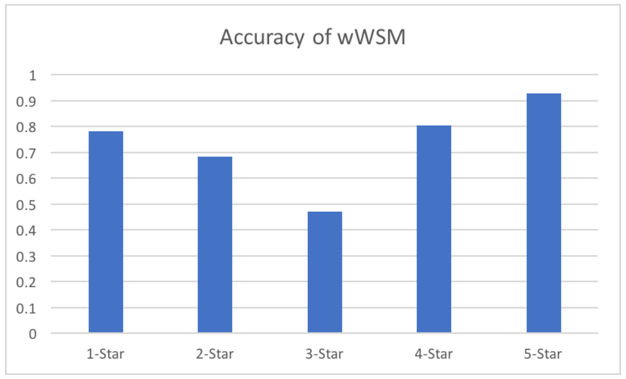

Fig. 3: Accuracy results for reviews with different star ratings using the weighted WSM scheme. Overall accuracy is about 75%.

TFIDF scheme. For instance, in F1 scores, the weighted WSM scheme achieves about 85%, compared to the 80% performance of Delta TFIDF.

TABLE V: Comparing our design with Delta TFIDF

Dataset	Alg.	FP	FN	TP	TN	F1
Bl'ed	wWSM	0.3707	0.1165	0.8835	0.6293	0.7842
Bl'ed	DT	0.3414	0.1503	0.8497	0.6586	0.7759
Unbl'ed	wWSM	0.3695	0.1198	0.8802	0.6305	0.8543
Unbl'ed	DT	0.9961	0.0010	0.9990	0.0039	0.8037

In Figure 3, we compare the Accuracy results for groups of reviews with different star ratings using the weighted WSM scheme. Therefore, all reviews with ratings of each of the 1-5 star ratings are grouped together for wWSM to analyze. Overall accuracy is about 75%. The accuracy for 3-star reviews is significantly lower than other reviews, indicating the vague meanings for such reviews. Therefore, a more accurate classification of 5 classes might be needed in order to raise the prediction accuracy for 3-star reviews in the wWSM scheme (although some changes are needed). We discuss this in our future research direction in Section V.

Algorithm complexity and runtime: Overall, the wWSM scheme has a complexity of $O(NT \log U)$, where U is the largest number of useful votes among any review, N is the number of reviews, and T is the number of terms/words in the longest review. Using a computer server equipped with an Intel Xeon E5-2430 v2 2.5 GHz (6 cores) and 16G memory running Ubuntu 16.04.5 LTS, it took about 23 minutes to pre-process (about half of which was removing words with frequencies less than 50) and 54 minutes to run our wWSM scheme. When we increased the size of our training dataset from 476,720 reviews to 605,000 reviews (an increase of 27%), the runtime of wWSM increased from 54 minutes to 68 minutes (an increase of 17%.) In comparison, the baseline Naive Bayes technique needed 55 hours.

V. Concluding Remarks

As social media evolve, user experience and comments have become increasingly important. In this work, we have proposed a weighted Word Sentiment Magnitude (wWSM) scheme that is able to help us quantify user sentiment through simple training and the results can be used to predict review rating of new comments rather accurately.

Most interestingly, the wWSM scheme shows great improvements in classifying 1- and 5-star reviews even in datasets with unbalanced class data. An overall accuracy of 75% can be achieved for all reviews and 92% for 5-star reviews.

Furthermore, these user responses could help us weight datasets to increase their quality. We can use techniques to handle the resulting non-evenly weighted dataset, such as the word sentiment magnitude test we have developed. This test, as other feature selection processes prove, shows that "quality" is better than "quantity" and that we can process data more quickly and accurately by using such methods.

In this research, we have not tested other related sentiment analysis schemes such as Support Vector Machine (SVM), decision tree, random forest even though these are strong candidates with better results. In our future work, we also plan to investigate a 3-class (positive/negative/neutral) or 5-class (one for each star) classification problem instead of a 2-class positive/negative classification problem.

References

[1] Y. Jo and A. H. Oh, "Aspect and sentiment unification model for online review analysis," in *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, ser. WSDM '11. New York, NY, USA: ACM, 2011, pp. 815–824. [Online]. Available: http://doi.acm.org/10.1145/1935826.1935932

[2] S. Tan, X. Cheng, Y. Wang, and H. Xu, "Adapting naive bayes to domain adaptation for sentiment analysis," in *Advances in Information Retrieval*, M. Boughanem, C. Berrut, J. Mothe, and C. Soule-Dupuy, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 337–349.

[3] B. Pang and L. Lee, "Opinion mining and sentiment analysis," *Foundations and Trends in Information Retrieval*, vol. 2, no. 1-2, pp. 1–135, 2008. [Online]. Available: http://dx.doi.org/10.1561/1500000011

[4] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, ser. HLT '11. Stroudsburg, PA, USA: Association for Computational Linguistics, 2011, pp. 142–150. [Online]. Available: http://dl.acm.org/citation.cfm?id=2002472.2002491

[5] W. Duan, Q. Cao, Y. Yu, and S. Levy, "Mining online user-generated content: Using sentiment analysis technique to study hotel service quality," in *2013 46th Hawaii International Conference on System Sciences*, Jan 2013, pp. 3119–3128.

[6] M. Salehan and D. J. Kim, "Predicting the performance of online consumer reviews: A sentiment mining approach to big data analytics," *Decision Support Systems*, vol. 81, pp. 30 – 40, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167923615002006

[7] L. Dey, S. Chakraborty, A. Biswas, B. Bose, and S. Tiwari, "Sentiment analysis of review datasets using naive bayes and k-nn classifier," *arXiv preprint arXiv:1610.09982*, 2016.

[8] S. Bakhshi, P. Kanuparthy, and D. A. Shamma, "Understanding online reviews: Funny, cool or useful?" in *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*. ACM, 2015, pp. 1270–1276.

[9] X. Lei, X. Qian, and G. Zhao, "Rating prediction based on social sentiment from textual reviews," *IEEE transactions on multimedia*, vol. 18, no. 9, pp. 1910–1921, 2016.

[10] A. K. Uysal, "An improved global feature selection scheme for text classification," *Expert systems with Applications*, vol. 43, pp. 82–92, 2016.

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

[12] J. Martineau and T. W. Finin, "Delta TFIDF: An improved feature space for sentiment analysis," in *ICWSM*, 2009.

[13] "Yelp challenge dataset," https://www.yelp.com/dataset, accessed: 2019-03-27.

[14] I. Feinerer and K. Hornik, *tm: Text Mining Package*, 2018, r package version 0.7-6. [Online]. Available: https://CRAN.R-project.org/package=tm

[15] S. Bird, E. Loper, and E. Klein, "Natural language processing with python," O'Reilly Media Inc., 2009.

[16] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2017. [Online]. Available: https://www.R-project.org/