Sentiment Analysis on IMDB Reviews Using Ensemble of SVM Models

Andrew Murwin
School of Computing and Augmented Intelligence
Arizona State University
Tempe, Arizona USA
amurwin@asu.edu

ABSTRACT

As social media sites become increasing popular, people's opinions online become more important than ever. As a result, the need for a way to accurately, automatically interpret opinion continues to grow. Semantic analysis fills this role by serving as a way for machines to interpret and categorize documents based on their sentiment. This research proposes a method for a different form of semantic analysis. Rather than apply a new model to the traditional approach of binary classification, this paper will seek to predict semantic meaning through multinominal classification. The proposed framework is used for sentiment analysis on a dataset of 50,000 reviews pulled from IMDB, with each one transformed into a word-based feature vector and labelled with the review's score. The data was then processed through eight different SVMs to generate an ensemble for classification. The trained ensemble achieved a final accuracy of 76.7% accuracy for all data, and an accuracy of 95.8% for classified data.

1 INTRODUCTION

As people, we are constantly sharing our opinions with others, and using other's opinions to inform our decisions. Outside of people we know personally, we also seek others' opinions online. We use the opinions of the masses to understand the public consensus on new content and ideas, and now with the use of social media, this becomes more common than ever. Just as we understand these ideas others present, we also want computers to properly interpret them [1]. To do this, tools like machine learning and natural language processing are used in sentiment analysis. Sentiment analysis is the technique of identifying subjective information that's holds opinionated information [2]. Generally, this information is categorized one of two ways. The first is a trinary system, with each document receiving a label of either positive, negative, or neutral. This is the type of data most used when semantic analysis frameworks are proposed [3][4]. The other method is to apply a rating scale, with the most common scales being 1-5 and 1-10, with one being the worst review and 10 being the best review. This is the format review websites like IMDB often store their reviews in [5].

Sentiment analysis into two components: preprocessing and model generation. The preprocessing has many different sub-methods, with only a portion being done in each sentiment analysis experiment. Part of Speech (POS) tagging is used to tag language grammar with its part of speech to preserve semantic relationships. "Bag of Words" consists of taking a document and converting it into a vector that represents the count of each word in the document [6]. This feature is either used on its own, or in calculating the Term Frequency (TF) or Term Frequency-Inverse Document Frequency (TF-IDF) statistics for a document [1]. Symbol removal removes symbols such as "?", "!" from the dataset to help consolidate the symbols and reduce the vector sizes. Stop word removal is done for a similar reason, removing words like "me", "you", "we" that hold neutral meaning to reduce the data size [7]. Sometimes, non-traditional words will be added to the stop word list, like "movie", should the context of the data deem it appropriate [8]. Lemmatization is the process of breaking down words into the stem components to group multiple forms of the same word. The second half of sentiment analysis, model generation, has can take many different forms. Success in binomial classification has been found with KNN [1], SVM [5] and LSTM [8], among other models. Some applications use an ensemble of multiple models to further improve performance [3][8].

Although there have been many successful implementations of binomial classification semantic analysis in the past, the literature regarding multinomial classification in semantic analysis is quite sparse. This paper proposes a method of classification for semantic analysis that will predict a document's sentiment on in the range [1-10], with one being extremely negative and 10 being extremely positive. The problem with current semantic analysis is that, while a document rated four is a negative rating, it is ultimately as close to a document rated one, which is also negative, as it is to a document rated seven, which is positive, yet when doing a binomial classification, it is deemed correct to group four with the ones, but not the sevens. Using an ensemble of SVMs, the method in this paper predicts the score of a document, and rate its accuracy based on how close it is to the true label, with it being more accurate the closer it is.

The structure of the rest of the paper is as follows. Section 2 covers a literature review of related works and how they approached similar classification issues in sentiment analysis. Section 3 covers the technical details of the methodology used.

Section 4 covers the results of the experiments performed. Section 5 provides room for discussion on the results, and section 6 concludes the paper.

2 RELATED WORK

This section focuses on previous works that used similar techniques in their models. Unfortunately, although there were many works on binomial classification of semantic analysis, not many works in multinomial classification were found in the literature, and thus most of the related works will focus on the former as many of the techniques are still relevant in multinomial classification.

One related work, as well as the origin of the dataset for this project, comes from Maas et al. [5]. Overall, the paper focused on two main properties: semantic similarities and word sentiment. Semantic similarities served to group reviews with similar intention without having any form of classification to put them against. Word sentiment served to provide sentimental value to words based on semantic similarities and was then used to predict document. It also took some unique approaches such as including traditionally excluded stop words (like "not") and including specific non-word characters indicative of sentiment like '!' and '?'. The work performed its tests on multiple datasets for comparison, including the IMDB dataset which they later released to serve as a benchmark for future research. When performing their binary classification on the IMDB dataset, the average performance was ~88% accuracy. While not directly comparable due to the difference between binomial and multinomial classification, many of the ideas such as bag of words and word sentiment will serve as a base for this paper.

Yasen et al. [9] proposed a similar line of thinking, but with the focus on testing different model types. The authors tested eight different model types on the IMDb dataset, which was processed with tokenization, word filtering, and Lemmatization to stem words and promote dataset unity. Out of the models tested, including SVM, KNN, and SGD linear classifier, the most successful model was Random Forest binary classification with an accuracy of 96%.

Kumar et al. [1] built further on this through additional preprocessing and including "lexical" statistics into the data provided into the models. This included the Term Frequency-Inverse Document Frequency, Positive and Negative Word Count, and Positive and Negative Connotation Count. The authors also chose to remove stop words from the dataset. The results of developing a model on the original data and the data + lexical statistics on four different models: SVM, Naïve Bayes, KNN, and Maximum Entropy, with an increase in accuracy shown across all four models.

Rathor et al. [3] proposed a framework of data preparation and machine learning techniques for classifying the data according to multiple models. The data cleaning techniques included the removal of punctuation and stop words, tokenization, and Lemmatization. The authors then proceeded to create an ensemble of three classifications: Logistic Regression, SVM, and Naïve Bayes, with accuracies of 75%, 58%, and 75% respectively. Once combined into an ensemble however, the proposed probabilistic model's accuracy was a significantly higher 98.7%.

Shaukat et al. [6] proposed a framework for classification based on word meaning and connections rather than one based solely on correlations. The authors used four key ideas to relate and categorize text: Bag of words, WordNet dictionaries, Part of Speech tags, and semantic relationships. The authors tested their data on a Multilayer Perceptron and achieved an accuracy of 91.9%.

Minaee et al. [8] proposed a framework to categorize multiple sources of sentiment data. The authors chose to exclude both standard stop words and context specific ones, such as "movie" and "film". The data was then tested on an ensemble of a CNN and a bidirectional-LSTM. On the IMDB dataset, the proposed ensemble had an accuracy of 90%. This framework in particular is interesting because, although the authors chose to perform a binary classification, they noted that the true output of the LSTM was a range of values in [0,1], which matches the objective of our paper's multinominal classification.

3 METHODS

The dataset used in the analysis is the 50,000 review IMDB movie review dataset (http://ai.stanford.edu/~amaas/data/sentiment/). In the provided dataset, the data is split between a train set and a test set, and between positive reviews and negative reviews within those sets. Additional metadata on the reviews in included, but not relevant for this paper. Each document is labeled with a unique ID and the rating given alongside it. The exact distribution of the classes in the dataset is in Table 1.

Table 1: IMDB Review Dataset Distribution

Rating	Train Count	Test Count	Total Count
1	5,100	5,022	10,122
2	2,284	2,302	4,586
3	2,420	2,541	4,961
4	2,696	2,635	5,331
7	2,496	2,307	4,803
8	3,009	2,850	5,859
9	2,263	2,344	4,607
10	4,732	4,999	9,731
Total	25,000	25,000	50,000

The original dataset is provided as two feature vector files and one vocab file. The feature vector in both the train and test splits combine both the positive and negatives into one file. Each of the feature vectors is a sparse vector stored in LIBSVM format, which is used for storing vectors that contain mostly zero values. The feature vectors store each review as a modified bag of words. In a

way similar to [6], each review is stored as a list of word, amount pairs as a modified version of bag of words, providing a compact list of all words used in each review. In a smaller scale bag of words, the records would be represented by a series of vectors, one per review, where the same index in each vector represents the count of the same world. This allows vectors to easily compare word counts without even having to know what the words are. sparse vectors, the two reviews are compared by seeing which indexes are stored in both, and which only exist in one. This is especially important as many words that only appear a few times would have an exponential effect on the space and time requirements of comparing vectors, with up to 179,000 elements being necessary for each comparison. In this paper's implementation of the ensemble classifier, the data will remain a 50/50 split between the train and test set as provided.

One disadvantage to this approach is that words lose contextual meaning. In many cases, the meaning of words is dependent on the context of the phrase it is a part of. The best example of this is negating words like "not". In the phrases "not good" and "not bad", the word "not" has the same effect, but different semantics, with the first phrase being negative and the second being positive. The way bag of words is implemented means that all these opposing views are consolidated into one interpretation that is consistent across all reviews. Ultimately, contextual words like "not" were still included as negating words still have semantic meaning regardless of the context. In the case of negating words, they have the effect of shifting the meaning towards a neutral stance.

The classifier used in this paper is an ensemble of eight independent SVM classifiers. For each classifier, one of the eight valid classes [1-4,7-10] are selected. A mask is then applied to the training data such that each instance that matches that class remains as that class, and all instances that don't become zero. An example would constitute applying a 2 filter to the list [1,2,3,4,5] and returning the list [0,2,0,0,0]. This process is repeated for the other seven models, such that eight binomial classifiers are generated based on this masked data. When predicting the test data, the same filter is applied to determine which scores are predicted for each sample.

Each of the eight models is then used to predict the entirety of the test set. The ensemble then predicts the class for each test data point as the average of the classes where the model for that class predicted the test data point to be in that class. Data points that were not predicted by any of the ensemble models were treated two different ways. In the standard approach, unclassified data points were scored as a 5.5, as a document that is scored as neither positive nor negative must be neutral. This data is used to provide an overall accuracy for the ensemble. The second approach is the remove all data points not classified by the model, and only use the remaining ones to determine the accuracy of the model. In this case, the accuracy calculation represented the effective confidence of the classification by removing data points that were not distinct.

Using this process, multiple kernels and regularization parameters were tested to determine the optimal combination for the data set. The two kernels used for classification, linear and RBF, were used to cover cases of both linear and non-linear differentiable cases. The five regularization parameters were 10, 1, 0.1, 0.01 and 0.001. These regularization parameters were used to test different decision boundaries to find an optimal one that is both not too loose with the data points its classifies and is not conducive to overfitting.

4 RESULTS

The accuracy scores for the models themselves will be calculated using the standard formula for accuracy of the number of correct predictions, both positive and negative, out of the total number of predictions. The accuracies for the individual models were relatively consistent across the 80 models run. Out of all the models, the individual model with the best performance is the linear SVM on class 2 with C=0.001. The model performed with an accuracy of 90.8%. Individually, the model with the worst performance is the linear SVM on class 10 with C=10. The model had an accuracy of only 78.8%. Overall, the average accuracy of models is 86.7%, with the linear accuracy averaging at 85.5%, and the RBF at 87.8%. Although the accuracy individual models performed well, the substantial number of true negatives in each dataset after the filters are applied means that accuracy does not give the full picture. Even so, accuracy is still used for consistency so the individual models could be compared to the ensembles which used a similar accuracy calculation to binomial accuracy.

Table 2: Individual Model Accuracies (C=1)

Class	Linear	RBF
1	0.81212	0.84972
2	0.84496	0.90792
3	0.83276	0.89844
4	0.83380	0.89460
7	0.85748	0.90776
8	0.81984	0.88600
9	0.84912	0.90624
10	0.78916	0.81532

The accuracy for an ensemble is calculated in a way that accounted for the similarity between the projected score and the true score. The formula is based on the binary classification accuracy formula, with the only modification being the inclusion of the 1/9 scaling factor to account for the range being [1-10] instead of [0-1]. The ensemble accuracy calculation uses the following formula:

$$\frac{\sum_{n=1}^{N} 1 - \frac{|y_n - \hat{y}_n|}{9}}{N}$$

Equation 1: SVM Ensemble Accuracy

In the equation, N is the total number of test data points, y_n is the true class for the nth data point, and $\hat{y}_n$ is the predicted class for the nth data point. The formula scales the true minus predicted value to a decimal based on the given class range, where the two furthest classes, 1 and 10, should give a value of 0. The value is then subtracted it from one to get a percentage which is then averaged over all the test data points.

Table 3: Accuracies including Unpredicted Data

C	Linear	RBF
10	0.76624	0.73884
1	0.76672	0.68397
0.1	0.76366	0.63447
0.01	0.72885	0.63446
0.001	0.67177	0.63446

Out of the ten ensembles that were run, the best is the linear ensemble with C=1, scoring an accuracy of 76.7% across all 25,000 data points in the test set. With the furthest parameters from it, the RBF ensemble with C=0.001 scored the worst with an accuracy of 63.4%. Over the five regularization values tested, the linear kernel ensembles showed a steady decrease in accuracy as the value decreased. Over those same five values, the RBF kernel accuracy drops immediately then plateaus at 63.4%.

In the table below, the accuracies were calculated using a modification of Equation 1. This modified equation calculates the accuracies of the models with the unpredicted data removed. Instead of the total number of data points, N represents the total number of data points classified by any of the models in the ensemble. Additionally, if there are no data points classified by the ensemble, the accuracy is determined to not exist rather than be defined as a default value.

Table 4: Accuracies with Unpredicted Data Removed

C	Linear	RBF
10	0.80580	0.91056
1	0.80766	0.95793
0.1	0.83301	0.88889
0.01	0.91119	N/A
0.001	0.95418	N/A

After removing data that none of the models in the ensemble classified, the models scored significantly higher accuracy. The C value also presents an inverse relationship with the linear ensemble accuracies, with a smaller C value generating a higher accuracy. This is the opposite when the unpredicted data is included, where smaller C values lead to smaller accuracies.

The RBF appears to have no direct correlation between itself the C value, making it the only one of the four graphs to lack a connection. Part of this is due to the lack of accuracies calculated. In the RBF values, an accuracy of N/A represents an accuracy in where there were no data points classified by any of the models in the ensemble. While calculating additional C values may provide a general trend, the three accuracies calculated indicated no evident pattern.

5 DISCUSSION

Overall, the ensembles generated did an adequate, although not outstanding, job performing sentiment analysis on movie reviews to determine the semantic meaning behind the reviews. The results at all steps of the analysis were informative as to the advantages and disadvantages of using a SVM ensemble in semantic analysis.

Although there is skew in the distribution of the dataset, with both classes one and ten having almost twice as many as any other class, the accuracies of the individual models, and by proxy the ensembles, did not suffer because of it. Because the multinomial classification problem is broken down into a series of binomial classification problems, the dataset is transformed each time, so the class's data is around 10-20% of the dataset, while the data that didn't fit the class filled in the rest. This, combined with the quantity of data provided, meant that there were not issues with the edges classes skewing the results of the ensemble.

Comparing the individual models and the ensemble models is informative in displaying where the ensemble struggled. The reason for using an ensemble in machine learning is to take multiple models that predict the same thing and decide on an outcome based on the individual model outputs. The ensemble implemented in this paper similar but differs in that the individual models predict independent data. This seems counterintuitive because all models in the ensemble predict the same dataset. Ultimately though, the original dataset had one label for each review, meaning that in a perfect prediction the data is split between the models, and there is no overlap in the prediction areas.

Traditionally, an ensemble is used to aggregate the predictions of multiple models and take an average approach response to improve the accuracy of the predictions. In this case, overlapping knowledge is a necessary bonus because is less susceptible to issues with individual models. In contrast, the ensemble in this paper uses multiple models to fill in knowledge gaps, forming an amalgamation instead of an average. In the case of this paper, overlapping knowledge is a necessary disadvantage because the classifications from each model in the ensemble are necessary, but cases where multiple models classify the same data point have a significant negative impact on the accuracy, especially if those classes are further apart.

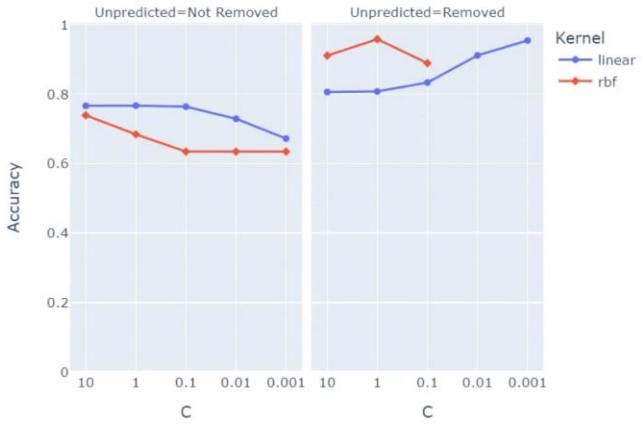

Figure 1: Accuracies of Ensemble Models

Within the accuracies themselves, there are multiple correlations that should be discussed. When looking that the accuracies of both the linear and RBF kernels, there is a general downward trend in the accuracy as the C value shrinks. As C decreases, the strength of the regularization increases which leads to overfitting. Due to the ensemble used, the decrease in C value causes each of the component models to overfit on their individual classes, an effect that compounds once the ensemble is introduced. This is where the issue of conflicting knowledge in the ensemble is most apparent and has the largest impact on the accuracies of the models.

Another prevalent occurrence in the data is the convergence of the RBF kernel ensemble. As the C value shrinks, the ensembles converge on an accuracy of 63.4%. Upon further examination into the individual model accuracies, this convergence occurred because the models underwent little change after C=0.1, and no change once C=0.01. In the case of the RBF ensemble, reaching convergence meant that none of the underlying models were predicting any of the classes for any of the models. This is confirmed when removing all the unpredicted data. As seen in Figure 1, the RBF graph has no accuracy for C=0.01 and C=0.001 as there are no data points to predict the accuracy of. The accuracy of 63.4% that the RBF ensembles kept reaching is the baseline accuracy calculated when each data point's predicted score is set to 5.5.

Interestingly, the correlation between the linear and removed unpredicted data is the inverse of the correlation between the linear and kept unpredicted data. The accuracy of the linear model is in line with the rest of the data. As C decreases, the regularization factor begins to cause overfitting. While overfitting would generally cause a drop in accuracy, what happens is the increasingly overfit models cause more data points to remain unpredicted and get dropped from this calculation. In return, the predictions certainty increases, as the only data points left predicted were the ones that the model can classify under tight overfitting.

This still leaves the issue of the RBF with removed data graph. Like the accuracy convergence issue, the RBF had issues with the lack of available data. In the case of the linear kernel, the progressive decrease in the amount of data led to gradual overfitting. In the case of the RBF however, the initial dataset with the unpredicted data removed is already significantly cut, to the point where further refining the dataset by decreasing the C value leads to a decrease in accuracy due to the lack of substantial data. This is supported by the almost immediate convergence shown by the data in Table 3, and the existence of N/A values in Table 4.

The frequency of use of the average value also has significant impact on the accuracies of the models themselves. When comparing the models and their relative ensembles, there is a distinct disconnect in the accuracies. In the case of the linear kernel, the accuracy went from 85.5% to 76.7% in the best case, an 8.8% decrease. Yet when compared to the best-case accuracy of 95.4% when the unpredicted data is removed, there is a 9.9% increase in accuracy. On data points successfully classified by any of the models in the ensemble, there is a significant improvement in accuracy through using the ensemble. The RBF shows a similar change. The individual models score an accuracy of 87.7% while the best-case ensemble without the data removed is 73.9%, a 13.8% decrease. In contrast, the best accuracy with the unpredicted data removed is 95.8%, an 8.1% increase. While RBF isn't as reliable as linear kernels due to its known small test dataset, the correlation between the two indicates the efficacy of the specialized models.

6 CONCLUSION

Over time, sentiment analysis has become a prevalent method for evaluating public sentiment towards different topics. It's potential for interpreting meaning makes it an asset in any person-related industry. Pulling from various ideas, I put together a framework for converting a previously binomial problem into a multinomial one. Rather than just determine whether a review is positive or negative, the framework can tell how positive or how negative the model is. Unfortunately, the complexity of human language made this task difficult. In this study, I used an ensemble of Support Vector Machines to classify movie reviews. Rather than look at the problem as a multinomial problem, it was split a part into eight binomial problems according to the eight possible classed the score of the review could be. Each one is evaluated with a SVM, and an ensemble is used to combine these results into a coherent score. The result was the creation of an ensembles with varying accuracy depending on the prediction task. The best ensemble in terms of the overall accuracy was a linear SVM with C=1, with an accuracy of 76.7%. The best ensemble in terms of the accuracy of data predicted by at least one model in the ensemble was a RBF SVM with C=1 and an accuracy of 95.8%.

While not the best performing model, it still turned out decent considering it is performing multinomial classification.

In the future, there are multiple approaches that used to improve the performance of the ensembles. One issue I ran into during model generation was that as C became larger, the time to generate the models grew exponentially, to the point where running 16 models took over ten hours. Part of the issue with this was the total possible number of words in each vector. While most reviews contained only a few hundred words, the total vocabulary for the dataset was over 89,000 words. Through additional preprocessing to remove stop-words and misspellings, and stem words through processes like Lemmatization, the vector size could have been decreased to improve run speeds and allow for the testing of larger regularization parameters.

Another further improvement would be to consider the use of alternate models. One model to investigate would be the multiclass Support Vector Machine, which can perform multinomial classification with just one model. One of the advantages to multiclass SVMs is that they can force a label on each test data point, which would prevent the issue of unclassified data that I had to deal with. In my proposed framework, I treated the data as labeled data, and while that is true, it also limited the options I considered. Rather than taking a categorical approach, I could have taken a continuous one. In my model, I turned scores into a continuous dataset by taking the average and using it as my selected ensemble score. Given that I was now working with continuous data, using a regression model designed for it may have provided more accurate scores.

Another further improvement could have been the use of additional models in the ensemble. One option would be to include both the linear and RBF models into a joint ensemble. Both predict correctly and confidently with a small enough regularization parameter, so combining all ten models may fill in more of the remaining gaps and further improve the accuracy. Additional supplementary models like the discussed multiclass SVM and regression models could also be included the ensemble to improve accuracy and guarantee a prediction on each test data point.

REFERENCES

[1] Kumar, K., Harish, B.S., and Darshan, H.K. 2019. Sentiment Analysis on IMDb Movie Reviews Using Hybrid Feature Extraction Method. International Journal of Interactive Multimedia and Artificial Intelligence 5, 109. http://doi.org/10.9781/ijimai.2018.12.005.

[2] Maria Giatsoglou, Manolis G. Vozalis, Konstantinos Diamantaras, Athena Vakali, George Sarigiannidis, and Konstantinos Ch. Chatzisavvas. 2017. Sentiment analysis leveraging emotions and word embeddings. Expert Systems with Applications 69, 214–224. https://doi.org/10.1016/j.eswa.2016.10.043

[3] Rathor, S. and Prakash, Y. 2022. Application of Machine Learning for Sentiment Analysis of Movies Using IMDB Rating. 2022 IEEE 11th International Conference on Communication Systems and Network Technologies (CSNT). http://dx.doi.org/10.1109/CSNT54456.2022.9787663.

[4] Abdalraouf Hassan and Ausif Mahmood. 2017. Deep Learning approach for sentiment analysis of short texts. 2017 3rd International Conference on Control, Automation and Robotics (ICCAR). https://doi.org/10.1109/iccar.2017.7942788

[5] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1 (HLT '11). Association for Computational Linguistics, USA, 142–150. Retrieved from https://ai.stanford.edu/~amaas/papers/wvSent_acl2011

[6] Shaukat, Z., Zulfiqar, A.A., Xiao, C., Azeem, M., and Mahmood, T. 2020. Sentiment analysis on IMDB using lexicon and neural networks. SN Applied Sciences 2. http://dx.doi.org/10.1007/s42452-019-1926-x.

[7] Saeed Mian Qaisar. 2020. Sentiment Analysis of IMDb Movie Reviews Using Long Short-Term Memory. 2020 2nd International Conference on Computer and Information Sciences (ICCIS). https://doi.org/10.1109/iccis49240.2020.9257657

[8] Shervin Minaee, Elham Azimi, AmirAli Abdolrashidi. 2019. Deep-Sentiment: Sentiment Analysis Using Ensemble of CNN and Bi-LSTM Models. arXiv:1904.04206. Retrieved from https://arxiv.org/pdf/1904.04206.

[9] Yasen, M. and Tedmori, S. 2019. Movies Reviews Sentiment Analysis and Classification. 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT). http://dx.doi.org/10.1109/JEEIT.2019.8717422.