Improving Sentiment Analysis Classification Scores Using Various Preprocessing Methods and More

Kim Lam
Department of Mathematics and Computer Science
Lawrence Technological University
Southfield, Michigan, USA
klam@ltu.edu

***Abstract*—Sentiment analysis is the field that is used to study emotion in informal forms of text. Twitter, Instagram replies, video comments, and Reddit posts are all examples of informal text that convey a user's emotions; analyzing the sentiment of these posts can be beneficial to gauge public opinions about products, movies, people, ideas, and more. In the rise of analyzing these informal texts, this introduces the question "What preprocessing methods are crucial for improving sentiment analysis?" For this study, a CRE Dataset with five classes of sentiment (anger, fear, joy, neural, and sadness) was provided with a baseline k-Nearest Neighbors (KNN) classifier. The baseline accuracy of the KNN classifier was about 38% without preprocessing. The overarching goal of this study is to increase the F1/accuracy of the given classifier, analyze the methods of doing so, and further research into other classifiers that could yield higher F1 score/accuracy. This study took over the course of about a month and a half. The proposed method of increasing the F1/accuracy of the KNN classifier includes (i) various preprocessing methods, (ii) applications of other classifiers, and (iii) parameter tuning. The report will cover in detail the related literature reviews and research done prior to the data analysis, the methods and design used in the study (dataset description, preprocessing methods, numerical representation, parameter tuning, and model building), the results and evaluation of the data analysis, future work to be done on the analysis of the dataset, and a conclusion and bibliography of the resources referenced in the study.**

***Index Terms*—data preprocessing, sentiment analysis, classifier, KNN, Decision Tree, Naive Bayes, MLP Neural Network, confusion matrix, classification report**

I. Introduction

Text mining is the field that processes text and extracts statistics to gain insight into its meaning. With the abundance of data that is provided online through social media [4], sentiment analysis is a popular way to help businesses, showbiz, movies, and sports industries [1] make data-driven decisions to benefit and improve their products/services [4]. In this study, a given dataset and baseline classifier will be used to answer the question of "What preprocessing methods are crucial for improving sentiment analysis?". Since this study is open ended, other research will be done to also look into other classifiers that may yield a higher F1 score/accuracy than the given classifier.

II. Related Work

Before delving into this study, some research was done on other scholarly papers on similar topics of sentiment analysis on informal text.

In the literature review of *Machine Learning-Based Sentiment Analysis for Twitter Accounts* [3] and *Deep Learning for Hate Speech Detection in Tweets* [8], the preprocessing methods for those datasets were much more involved since the goal of those studies were to compare the sentiment between two political candidates or detection of hate speech. There were similar classifiers and preprocessing methods used in this study compared to those two reports (such as the use of TF-IDF representation, Naive Bayes, and Decision Trees), but since this study is more open ended, not as much rigorous preprocessing will be done since the dataset doe not have as much noise as random Twitter posts.

The most important literature reviews that was found is on *MELD: A Multimodal Multi-Party Dataset for Emotion Recognition in Conversations* [9] and *Study of Twitter Sentiment Analysis using Machine Learning Algorithms on Python* [11]. The first dataset and preprocessing is a much different compared to the one used in this study - as it uses text, audio, and video cues to determine sentiment in dialogue in the TV series *Friends* - but still has some very valuable insights into sentiment analysis. In most informal forms of text, non-lexical utterances (e.g.: yeah, huh, uh, ah, etc.) are removed since they are viewed as noise to the other words that hold sentiment data. However, [9] holds a good point in that even non-lexical utterances actually do hold significant sentiment data (e.g.: "AHH!" can signify horror/surprise or "Whohooo!" can signify joy). Another non-lexical form of text to be considered is slang, as discussed in [11]. Slang can often be non-English words or references that contain sentiment behind them, so it is an important factor to consider when preprocessing. In the dataset for this study, however, this is not as prevalent.

From this research and literature reviews, the proposed method of increasing F1 scores and accuracy of classifiers as a whole would be (i) five main preprocessing methods of (a) lowercase conversion, (b) punctuation removal, (c), stop word removal, (d) number removal, and (e) removal of words of length 2 or less, (ii) parameter tuning of different classifiers, and (iii) the application of four different classifiers of (a) KNN, (b) Naive Bayes, (c) Decision Tree, and (d) Multi-Layer Processing (MLP) Neural Network.

III. Methods and Design

In this study, three major methods were applied to the CRE Dataset for sentiment analysis: preprocessing, application of different classifications, and parameter tuning.

A. Dataset Description

The dataset given for this study is a collection of sentences that have differing sentiment. The 5 class labels given are anger, fear, joy, neural, and sadness. This dataset is similar to the datasets seen in sources [6] and [7] where those datasets were also labeled with various sentiment labels.

B. Data Preprocessing

In this study, five preprocessing methods were used: (i) lowercase conversion, (ii) punctuation removal, (iii), stop word removal, (iv) number removal, and (v) removal of words of length 2 or less.

The first method of lowercase conversion is one of the standard preprocessing methods done in sentiment analysis. As later discussed in this study, this was done since the content of the dataset did not heavily depend on capitalization to convey sentiment.

The second method of punctuation removal is also a standard preprocessing method done in sentiment analysis. One caveat to keep in mind, is that multiple uses of punctuation can also convey significant data for sentiment analysis; i.e.: ”...” could convey sadness/apathy or ”!?!” could convey surprise or anger. In this dataset, however, punctuation was not a heavy indicator of sentiment so it was used to remove extra noise from the dataset.

The third method of stop word removal is the most fundamental preprocessing method that is used in sentiment analysis. This method is almost always a ”must do” in text preprocessing in which it removes words with no sentimental data (e.g.: ”a”, ”the”, ”to”, etc.). This was the most impactful preprocessing method that increased F1 scores/accuracy across all classifiers used in this study.

The fourth method of number removal is a preprocessing method that removes numbers from text. In this case, number removal was done since numbers do not usually have an impact on what kind of emotion is being conveyed.

The fifth method of removing words of length 2 or less is a crucial method to discuss. This preprocessing method is generally a good method to rid the data of extra noise that does not affect sentiment, however in this study it greatly decreased the F1 score/accuracy of some classifiers. Ultimately, this preprocessing method was not included in the final evaluations of each classifier.

C. Numerical Representation

The numerical representation utilized in this study was TF-IDF vectorization. TF-IDF is usually a good standard representation over Bag of Words (BoW) and was also used as in the given baseline KNN classifier, so this aspect was kept as a constant variable in the study. After vectorization of the preprocessed data, four classifiers were applied to the preprocessed dataset: KNN, Naive Bayes, Decision Tree, and MLP Neural Network.

Linguistic inquiry word count (LIWC) [3] or word embeddings could have been used as well, but to match the representation from the baseline classifier given, these methods were not used. In later studies, these methods could be explored further.

D. Parameter Tuning

On the aforementioned classifications, some parameter tuning was done to each model to achieve the best possible F1 scores/accuracy.

For the first classification of KNN, there were two parameters that were modified to get the best possible F1 Score and accuracy: max_features and n_neighbors. The max_features of the TF-IDF vectorizer was increased from 300 words from the baseline to 600 words. This increase gave the classifier a better sample to train off of. Several values were tested to prevent over fitting the data, and 600 words was the approximate best value for the max_features parameter. The n_neighbors parameter was adjusted to 3 based on value testing as well as plotting the Miss-classification errors against the K number of neighbors as seen in Fig. 1.

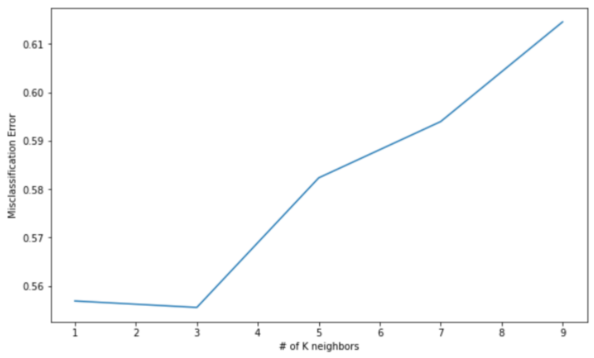


Fig. 1. KNN plot of error vs number of K neighbors.

For the second classification of Naive Bayes, the max_features parameter was modified. Starting with 100 words in the TF-IDF vectorizer, this resulted with an accuracy of about 40% accuracy. This was only a little better than the given baseline accuracy of about 38%. The max_features was eventually increased to 600 words to get an accuracy of about 52%. Over fitting occurred when the max_features went over 600 words, so 600 was determined to be the best value for this model.

For the third classification of Decision Tree, the max_features and max_depth parameters were modified. With several test runs, the best accuracy resulted in the elimination of both of these parameters. By eliminating the max_features and max_depth parameters, this allowed total freedom for the decision tree to train with all of the TF-IDF vectorized words with no limitations on the depth of the tree. With these limitations lifted from the tree, the accuracy resulted in a 60% accuracy. The following image shows the overall decision tree visualization in Fig. 2. The visualization shows the removal

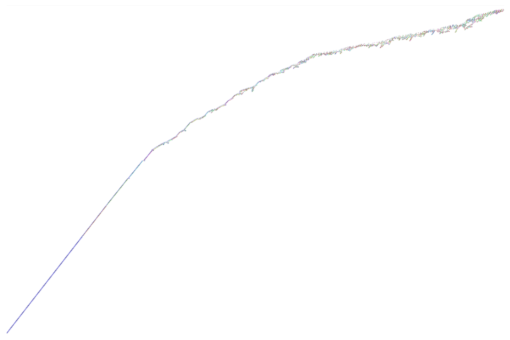


Fig. 2. Visualization of Decision Tree.

of depth limitation and results in a very large tree with a large depth.

For the fourth classification of MLP Neural Network, the max_features parameter was modified. Similar to the other models, the TF-IDF vectorizer features needed to be increased to increase the accuracy of the model. For the MLP Neural Network model, the max_features was increased to 600 words to obtain an accuracy of about 60%. Anything over 600 features resulted in over fitting and anything under would result in under fitting, so 600 features was chosen as the most optimal value.

E. Model building

In each evaluation of each classification, different models were build for each corresponding classifier with the python ".fit" functions.

For the KNN model, after the TF-IDF numerical representation was applied to the preprocessed data, the model built for this classification used the Euclidean distance to determine k most similar instances. Equation 1 for Euclidean distance is given below:

$$distance(x,y) = \sqrt{(x_1 - y_1)^2 + ... + (x_m - y_m)^2} \qquad (1)$$

For the Naive Bayes classifier, this model was created after the TF-IDF vectorization. Naive Bayes utilizes Naive Bayes Theorem 2, and, for the model used in the study, the Gaussian normal distribution 3.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \qquad (2)$$

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} \qquad (3)$$

For the Decision Tree classifier, the model was contracted using entropy 4 after the TF-IDF vectorization.

$$Entropy = \sum_j p_j log_2 p_j \qquad (4)$$

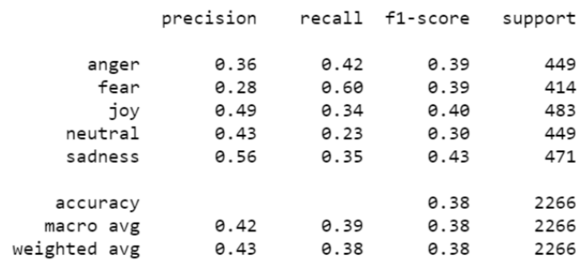
In our model, entropy was used since there was a high entropy (close to 1, which is the maximum impurity in the training set) and made for a very good training set. As seen later in the results, this entropy model resulted in the highest accuracy.

For the MLP Neural Network classifier, as indicated by the name of the classifier, multiple layers were used in the model building. An alpha (regularization term) and multiple hidden layers and neurons were used. The computation of neural networks uses weights 5 to train itself to output probabilities to then predict samples after training.

$$E(w) = \frac{1}{2} \sum_{i=1}^{N} (y_i - \hat{y_i})^2 \qquad (5)$$

IV. Results and Evaluation

The results varied from classification to classification depending on what preprocessing methods were applied and what parameters were tuned. Below is the classification report of the baseline KNN model that was provided with the non-preprocessed data in Fig. 3.



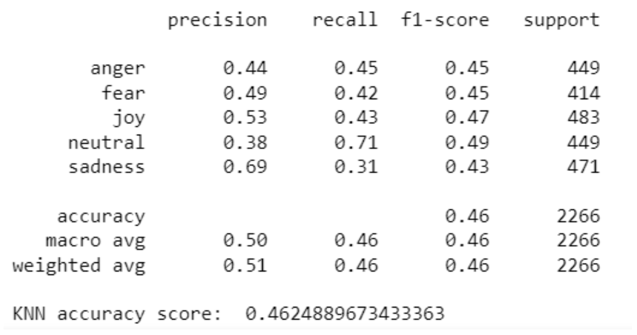
```
              precision    recall  f1-score   support

       anger       0.36      0.42      0.39       449
        fear       0.28      0.60      0.39       414
         joy       0.49      0.34      0.40       483
     neutral       0.43      0.23      0.30       449
     sadness       0.56      0.35      0.43       471

    accuracy                           0.38      2266
   macro avg       0.42      0.39      0.38      2266
weighted avg       0.43      0.38      0.38      2266
```

Fig. 3. Classification report for KNN before preprocessing.

For the first re-evaluation of the given baseline classification of KNN after preprocessing, the best accuracy obtained from the 4 methods of preprocessing (excluding removal of words of length 2 or less) was about 46% as seen in Fig. 4. This was an 8% increase of accuracy, but is not a good enough accuracy in predicting the sentiment of sentences. The accuracy is below 50%, so not even half of the predictions were correct. A confusion matrix was plotted to create a visualization of the correctly predicted sentiment labels versus the misclassification of the incorrectly predicted labels as seen in Fig. 5. The diagonal tiles from the upper left corner to the lower right corner show the correct classifications of the five sentiment labels. The more yellow a tile is, the higher classifications are made in that tile; similarly, the more dark



```
              precision    recall  f1-score   support

       anger       0.44      0.45      0.45       449
        fear       0.49      0.42      0.45       414
         joy       0.53      0.43      0.47       483
     neutral       0.38      0.71      0.49       449
     sadness       0.69      0.31      0.43       471

    accuracy                           0.46      2266
   macro avg       0.50      0.46      0.46      2266
weighted avg       0.51      0.46      0.46      2266

KNN accuracy score:  0.4624889673433363
```

Fig. 4. Classification report for KNN after preprocessing.

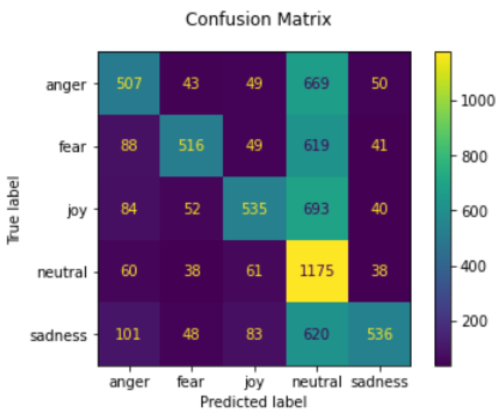
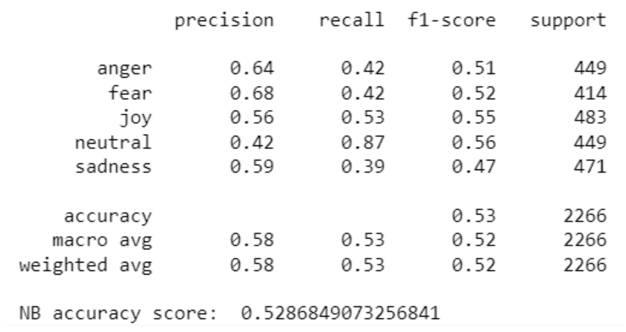


Fig. 5. Confusion Matrix for KNN.

indigo a tile is, the less classifications are made in that tile. The accuracy for KNN is low due to the constant misclassification of predicted neutral sentiment labels when those should have been predicted as one of the other four sentiment labels. The light turquoise tiles in the fourth column show that there was a high classifications for a lot of neutral sentiment, but the only true neutral labels (the correct classification of true and predicted labels) can be seen in the yellow tile in the same column. The other four columns show that there was a much smaller misclassification errors (as denoted by the dark indigo tiles) of the other four sentiments of anger, fear, joy, and sadness.

For the second classification of Naive Bayes, the best accuracy obtained from the 4 methods of preprocessing (excluding removal of words of length 2 or less) was about 52% as seen in Fig. 6. This model had a much better accuracy compared


```
              precision    recall  f1-score   support

       anger       0.64      0.42      0.51       449
        fear       0.68      0.42      0.52       414
         joy       0.56      0.53      0.55       483
     neutral       0.42      0.87      0.56       449
     sadness       0.59      0.39      0.47       471

    accuracy                           0.53      2266
   macro avg       0.58      0.53      0.52      2266
weighted avg       0.58      0.53      0.52      2266

NB accuracy score:  0.5286849073256841
```

Fig. 6. Classification report for Naive Bayes after preprocessing.

to the baseline with about an 14% increase. Comparing the re-evaluated KNN confusion matrix with the Naive Bayes confusion matrix, the misclassification errors occur much less as seen in Fig. 7. The light blue tiles seen around the upper-left and lower-right diagonals indicate that there is a much lower misclassification of the sentiment labels. However, the same trend from the KNN can be seen where the Naive Bayes model also had a difficult time in correctly predicting the neutral sentiment label.

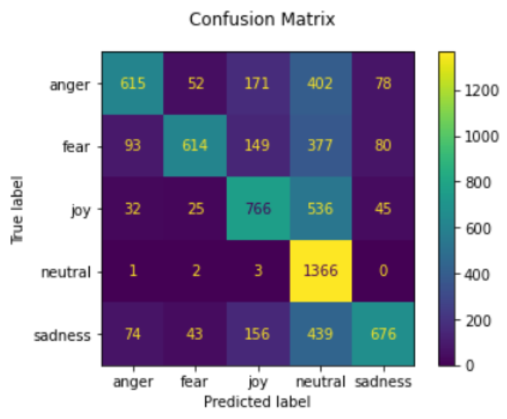
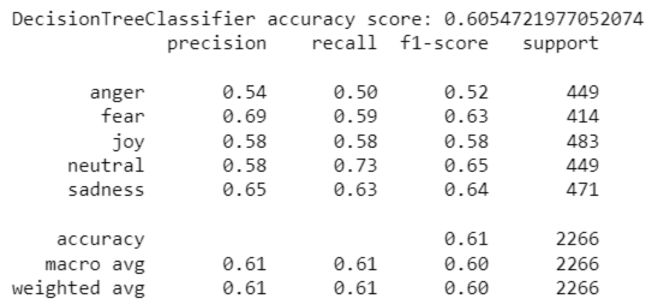


Fig. 7. Confusion Matrix for Naive Bayes.

For the third classification of Decision Tree, the best accuracy obtained from the 4 methods of preprocessing (excluding removal of words of length 2 or less) was about 61% as seen in Fig. 8. This classification yielded an even higher accuracy


```
DecisionTreeClassifier accuracy score: 0.6054721977052074
              precision    recall  f1-score   support

       anger       0.54      0.50      0.52       449
        fear       0.69      0.59      0.63       414
         joy       0.58      0.58      0.58       483
     neutral       0.58      0.73      0.65       449
     sadness       0.65      0.63      0.64       471

    accuracy                           0.61      2266
   macro avg       0.61      0.61      0.60      2266
weighted avg       0.61      0.61      0.60      2266
```

Fig. 8. Classification report for Decision Tree after preprocessing.

than the previous two models, with about a 22-23% increase in accuracy. This is a great accuracy to work with so far, and by comparing the confusion matrix in Fig. 9 with the previous two matrices, it is evident in why it is a better classifier. The

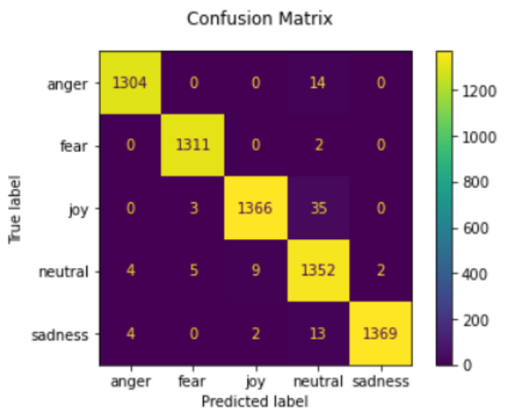
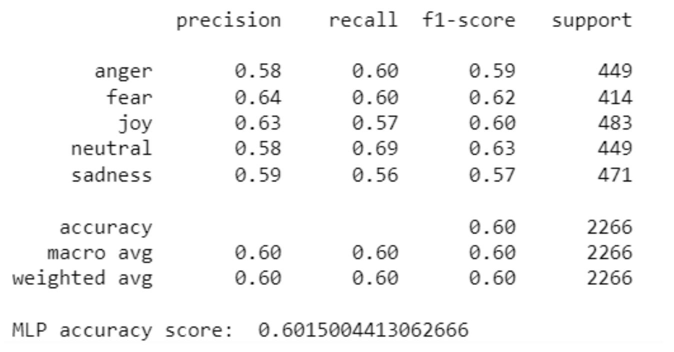


Fig. 9. Confusion Matrix for Decision Tree.

confusion matrix for the Decision Tree classifier has the least misclassification errors in predicting the sentiment of each sentence. There are little errors in misclassifications as seen by the dark indigo tiles around the bright yellow diagonal tiles that show the correct sentiment classifications.

For the fourth classification of MLP Neural Network, the best accuracy obtained from the 4 methods of preprocessing (excluding removal of words of length 2 or less) was about 60% as seen in Fig. 10. The MLP Neural Network



```
              precision    recall  f1-score   support

       anger       0.58      0.60      0.59       449
        fear       0.64      0.60      0.62       414
         joy       0.63      0.57      0.60       483
     neutral       0.58      0.69      0.63       449
     sadness       0.59      0.56      0.57       471

    accuracy                           0.60      2266
   macro avg       0.60      0.60      0.60      2266
weighted avg       0.60      0.60      0.60      2266

MLP accuracy score:  0.6015004413062666
```

Fig. 10. Classification report for MLP Neural Network after preprocessing.

classification accuracy is much better out of the first two classifications, but just slightly less than the Decision Tree classifier. To see why the MLP Neural Network classifier has a slightly lower accuracy, the MLP confusion matrix was plotted to be compared with the Decision Tree confusion matrix as seen in Fig. 11. The misclassification errors in

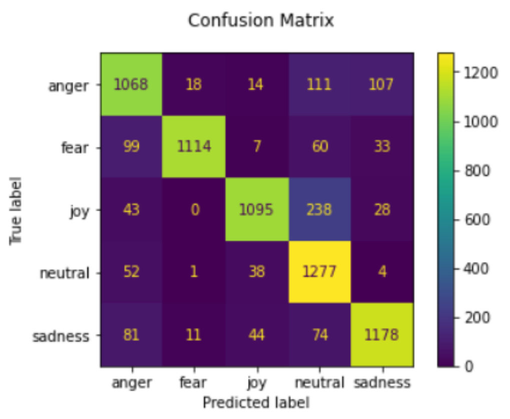


Fig. 11. Confusion Matrix for MLP Neural Network.

predicting each sentiment label is quite low, with most tiles surrounding the left-right diagonal colored dark indigo or indigo. The difference is that compared to the Decision Tree confusion matrix, it has a slightly higher misclassification error in predicting neutral sentiment labels and some sadness sentiment labels.

The best classifier out of the four created for this study is the Decision Tree Classifier with an accuracy of 61%. The following Table I is a tabular view of all of the results from each classification.

TABLE I
CLASSIFICATION F1 SCORES AND ACCURACY

Classification	**F1 Score**	**Accuracy**
KNN	0.46	46.2489%
Naive Bayes	0.52	52.8685%
Decision Tree	0.60	60.5472%
MLP Neural Network	0.60	60.1500%

V. FUTURE WORK

After analysis of this dataset and the evaluation of the results from this study, there is considerations to be taken into account that can lead to future work on not only this data, but all text data in general.

Preprocessing methods should be considered only once the dataset is closely studied. There are some vital methods that should be applied to text data in general, such as stop word and number removal. These removals allow for the words that are ”noise”, or have no sentiment value, be removed from the dataset. However, other preprocessing methods should only be applied if your dataset or your objective calls for it. For example, as seen in the dataset from [1], there are valuable non-word characters, such as emoticons, and punctuation usage that add to the sentiment of a sentence. These should not be removed if these elements are prevalent in a dataset, as non-character and punctuation removal can remove valuable sentiment. Conversion to lowercase can also be detrimental in removing sentiment; as internet culture is ever evolving, different types of sentiment can be expressed through text in all capital letters to express anger or excitement, while sarcasm can be expressed through the random capitalization of letters in a word (e.g.: iM SoOOooO sOrRy).

Another aspect to consider is the misclassification of neutral sentiment; this may be attributed that neutral sentiment may only be made up of words that are stop words or other words that are preprocessed out of the data. Keeping this in mind, preprocessing methods must be scrutinized even more to ensure that the study is correctly tested.

Finally, further research into other classifiers can also be done to see if they result in higher F1 scores/accuracy such as SVM (as seen in [5]), ensemble methods, random forests (as seen in [12]), and more. Tools made for sentiment analysis should also be considered, such as Rapid Miner, WEKA (as seen in [2]), Orange, MILAM (as seen in [10]), and more.

VI. CONCLUSION

Referring back to the question answered in this study, ”What preprocessing methods are crucial for improving sentiment analysis?”, the most beneficial/essential preprocessing methods for sentiment analysis is stop word and numerical removal. As these two preprocessing methods simply get rid of the extra noise in the data, other preprocessing methods should only be considered according to what your data/goals call for.

Further discoveries in this study revealed that for this dataset, Decision Trees was the best classifier out of the four tested with a 61% accuracy with minimal misclassifications.

References

[1] M. Javed and S. Kamal, Normalization of Unstructured and Informal Text in Sentiment Analysis, International Journal of Advanced Computer Science and Application, 2018.

[2] P. Baid, A. Gupta, and N. Chaplot, Sentiment Analysis of Movie Reviews using Machine Learning Techniques, International Journal of Computer Application, 2017.

[3] A. Hasan, S. Moin, A. Karim, and S. Shamshirband, Machine Learning-Based Sentiment Analysis for Twitter Accounts, Mathematical and Computational Applications, 2018.

[4] S. Pradha, M. N. Halgamuge, and N. Tran Qouc Vinh, Effective Text Data Preprocessing Technique for Sentiment Analysis in Social Media Data, IEEE, 2021. (CHECK year)

[5] K. Lavanya and C. Deisy, Twitter Sentiment Analysis Using Multi-Class SVM, International Conference on Intelligent Computing and Control (I2C2'17, 2017.

[6] D. Ghazi, D. Inkpen, and S. Szpakowicz, Detecting Emotion Stimuli in Emotion-Bearing Sentences, University of Ottawa, 2015.

[7] Y. Li, H. Su, X. Shen, W. Li, Z. Cao, S. Niu, DailyDialog: A Manually Labelled Multi-turn Dialogue Dataset, Hong Kong Polytechnic University, Chinese Academy of Science, and Saarland University, 2017.

[8] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, Deep Learning for Hate Speech Detection in Tweets, IIIT-H and Microsoft, 2017.

[9] S. Poria, D. Hazarika, N. Majumder, G. Naik, E. Cambria, and R. Mihalcea, MELD: A Multimodal Multi-Party Dataset for Emotion Recognition in Conversations, SUTD, National University of Singapore, Instituto Politecnico Nacional, Nanyang Technological University, and University of Michigan, 2019.

[10] K. Wang and X. Wan, Sentiment Analysis of Peer Review Texts of Scholarly Papers, Peking University, 2018.

[11] B. Gupta, M. Negi, K. Vishwakarma, G. Rawat, and P. Badhani, Study of Twitter Sentiment Analysis using Machine Learning Algorithms on Python, International Journal of Computer Applications, 2017.

[12] X. Fang and J. Zhan, Sentiment analysis using product review data, Journal of Big Data, 2015.