Machine Learning for Sentiment Analysis - Progress Report

Avery E. Peiffer
ECE, University of Pittsburgh
Pittsburgh, PA, USA
aep65@pitt.edu

Daniel C. Stumpp
ECE, University of Pittsburgh
Pittsburgh, PA, USA
dcs98@pitt.edu

Abstract

Emotional sentiment analysis can be used in a wide range of of applications, from gauging public opinion to studying the impact of social media on mental health. Despite the ease with which humans can perform sentiment analysis, the task poses a more substantial challenge for computers. We propose an investigation of state-of-the-art transformer-based machine learning methods to address this challenging task. We will also present an alternative solution based on more traditional machine learning techniques to evaluate performance against the state-of-the-art in both accuracy and inference time. Results and findings will be recorded in a final report.

Index Terms

Machine learning, sentiment analysis, natural language processing, transformers, support vector machines, long short term memory

I. Problem Description

Humans possess an innate ability to determine emotion from conversation without explicit cues. We automatically collect samples of a speaker's speech patterns, body language, choice of vocabulary, and the surrounding context, for example, as data points from which we construct an overall picture of their emotion. This subconscious analysis then informs our decisions and actions as we interact with those around us. While this task is incredibly simple for humans, determining emotional sentiment is much harder for a computer.

Sentiment analysis is a branch of computer science that aims to use machine learning techniques to automatically determine emotion conveyed through text. This proposed work will explore sentiment analysis using Twitter status updates (tweets) as samples for evaluation. The problem being addressed, proposed solution, related work, and project timeline will be outlined in the following sections.

II. Related Work

The following section will be a more detailed review of the work highlighted in our initial project proposal, as well as a further exploration of work in the space of NLP and sentiment analysis.

A. Methods for text feature extraction

In order for text data to be processed with machine learning algorithms, it must be distilled from a list of words into a set of workable features. This section provides a survey of various methods for extracting features from text, including traditional methods and newer, graph-based methods.

1) Traditional methods: Waykole and Thakare [1] conducted a review of three traditional methods for extracting features from text. The simplest method is the bag of words (BoW) method, in which each word in a block of text is compared to a lexicon of known words, and its frequency in the text data is counted. However, this method fails to identify words that are unique and important to a specific document, as their frequency counts are dwarfed by common words that make up the basis of a language. To mitigate this issue, each word can instead by assigned a Term Frequency - Inverse Document Frequency (TF-IDF) score. The term frequency is the frequency of the word in the existing document, and the inverse document frequency is the score of the word among all documents. This allows words that are important and unique to specific documents to be scored correctly.

The most complicated traditional method is Word2Vec, a series of two-layer neural networks that represent semantic contexts of words. It maps an entire corpus of text to a high-dimensional vector space, in which words with common contexts occupy similar locations in the vector space.

2) Graph methods: The authors of [2] introduced a graph-based algorithm to create an emotion-rich representation of text for emotion recognition tasks. An emotion graph was created using objective and subjective tweets scraped from the Twitter API. Network metrics were calculated for this graph, such as the eigenvector centrality and clustering coefficient of each vertex (which represent the tokens from the overall corpus). Vertices with high values in either of these metrics were used as candidates for a bootstrapping process to generate patterns of opinionated words. The authors then trained a neural network on pre-trained word embeddings from [3] to compare to the bootstrapped patterns. This process resulted in only emotion-rich patterns; that is, patterns that are present throughout the graph dataset and the pre-trained word embeddings. These patterns were then evaluated on several emotion recognition tasks and outperformed existing state-of-the-art techniques.

B. Basic classification techniques for sentiment analysis

Neethu and Rajasree [4] performed a general survey of classification techniques using a dataset of tweets about electronic products. The dataset was proprocessed to remove misspellings and slang words. For each sample in the dataset, a feature vector was created by extracting the positive and negative keywords from the tweet, including emoticons and hashtags. The authors then used three different classification techniques on their dataset for sentiment classification: naive Bayes, SVM, and maximum entropy. Finally, an ensemble classifier was created using the three previous techniques as base classifiers. All four methods had similar accuracies, ranging around 90%, when evaluated on the test dataset.

Rathi et al. [5] performed a similar evaluation of sentiment analysis using three classifiers: SVM, decision tree, and Adaboost decision tree. This work used three large and robust datasets, including the Stanford Sentiment140 dataset, containing over 1.6 million tweets [6]. The authors observed experimental accuracies of 82%, 67%, and 84%, respectively.

C. CNN and LSTM architectures for sentiment analysis

Sosa [7] introduced a combined LSTM-CNN architecture for sentiment analysis on Twitter data. They trained and validated their models on a large dataset compiled from a University of Michigan Kaggle competition [8] and Neik Sanders' Twitter Sentiment corpus [9], containing over 1.5 million total tweets labeled as positive or negative. The LSTM-CNN architecture used is shown below in Figure 1. The combined LSTM-CNN architecture outperformed both a LSTM and a CNN architecture individually, achieving an average accuracy rate of 75.2%, compared to 66.7% for the CNN architecture and 72.5% for the LSTM architecture.

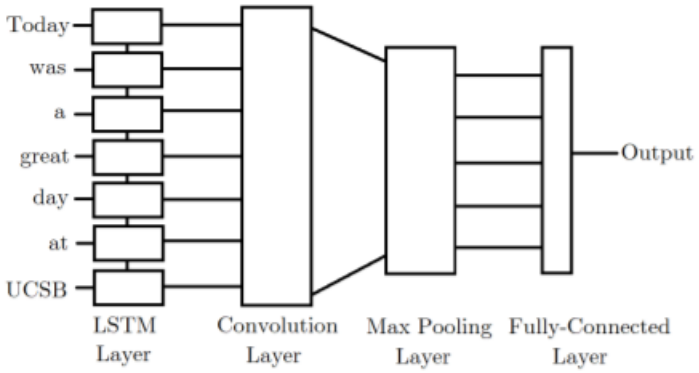


Fig. 1: Sosa's LSTM-CNN architecture for sentiment analysis

Chen and Wang [10] built on Sosa's implementation by adding an encoder/decoder framework to the existing LSTM-CNN architecture. The CNN is regarded as the encoder, corresponding to a two-layer deconvolution layer as a decoder. The architecture is shown below in Figure 2. The authors state that this structure allows the CNN to learn features more intrinsically and effectively. On the same datasets, the authors were able to achieve an accuracy of 78.6%.

D. Transformers

The architectures discussed in previous sections rely heavily on recurrent or convolutional neural networks that use an encoder and a decoder. The complexity of these models results in a high training cost, in the range of 10^{20} floating point operations per second (FLOPs). Vaswani et al. [11] introduced a much simpler architecture called the transformer, which does not need recurrence or convolution. The transformer architecture is shown below in Figure 3. This architecture relies solely on attention mechanisms [12], which are designed to focus on critical data while ignoring the rest. The transformer architecture improved on existing results for the WMT 2014 English-to-German translation task at a fraction of the training cost. The authors also showed that this architecture can be applied to English constituency parsing, which breaks down sentences into their syntactic structures, proving its worth in general NLP tasks.

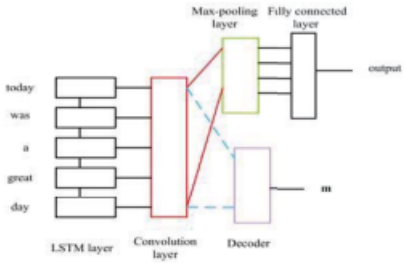


Fig. 2: Chen's and Wang's LSTM-CNN-encoder-decoder architecture, improving upon Sosa's sentiment analysis accuracy

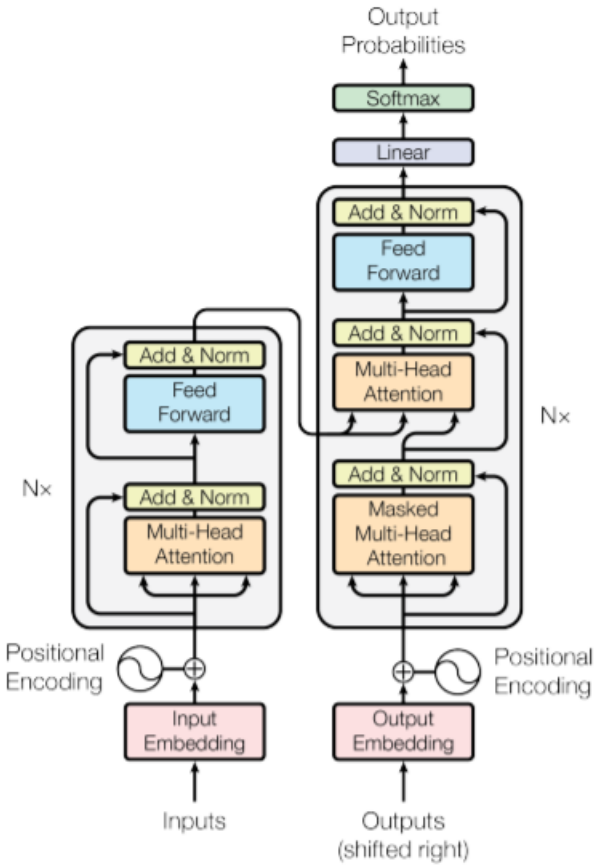


Fig. 3: The transformer architecture presented by Vaswani et al.

III. Project Progress

In this section we outline the project progress made thus far. We include analysis of the dataset being used along with preliminary results using traditional and more advance machine learning methods. Our project progress also includes the thorough literature review outlined in the previous section, which was essential to project decisions and progress.

A. Dataset Analysis

This section outlines a preliminary analysis of the tweet dataset being used. Table I outlines the basic information regarding the dataset. The dataset includes train, test, and validation splits, which represent 80%, 10%, and 10% of the samples, respectively. For models or methods that do not require cross validation, the validation set will be combined with the training set to represent a 90-10 train-test split. The total number of samples in the dataset is 20,000.

To effectively use the dataset, it is important to understand the distribution of classes. Fig 4 shows that the distribution of class labels is not balanced. The unbalanced nature of the dataset will be important when considering how to quantify model

TABLE I: Tweet dataset information and statistics.

statistic	value
classes	{sadness, joy, fear, anger, love, surprise}
splits	{train, val, test}
train samples	16000
val samples	2000
test samples	2000

performance as work progresses. Joy is the most prevalent class label with nearly 7000 instances. Sadness is the second most common class at just under 6000 instances while all other classes occur less than 3000 times.

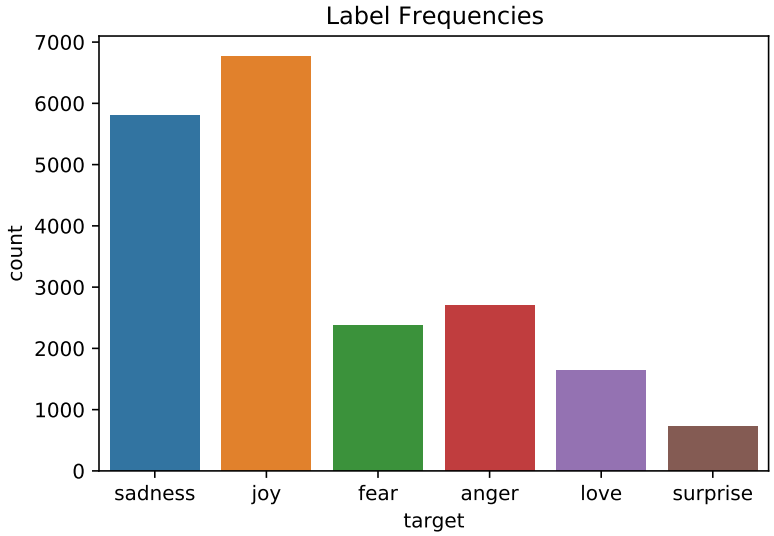
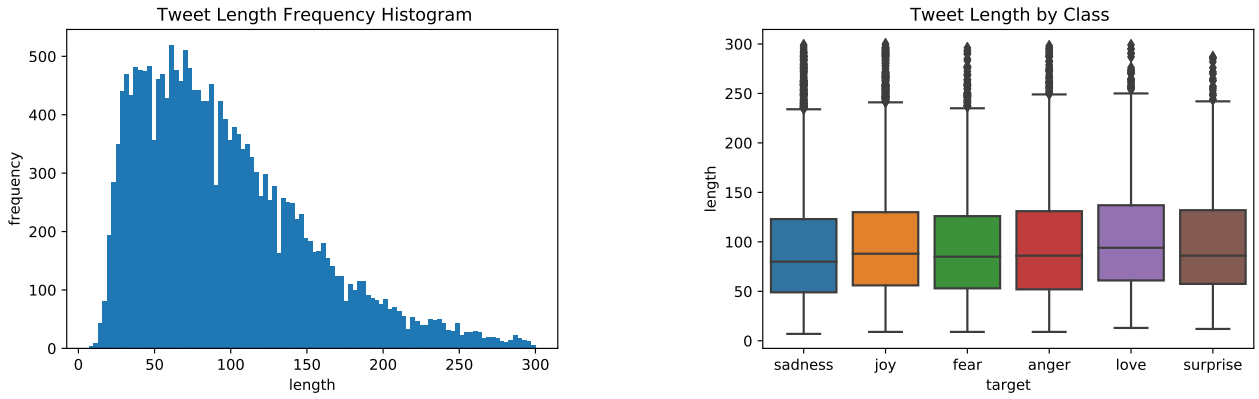


Fig. 4: Dataset label frequencies. The plot shows the unbalanced nature of the dataset being used.

Another characteristic of the dataset considered is the character length of the tweets. Fig. 5(a) shows a histogram distribution of the tweet character lengths in the dataset. The mean tweet length is 96.7 characters. The plot in Fig. 5(b) shows box plots for tweet length for each different class label. From this data we can see that the tweet length appears to have the same characteristics regardless of the tweet class. This indicates that tweet length may not be a salient feature for classification of tweets, however, this data will be considered as the project moves forward.

The last characteristic of the dataset initially considered is the word distribution in each class. Specifically, we are interested in seeing what words have high frequency in one class but not in the others. To get a sense for the what key words are in each of the classes, the top 15 unique words from each class were plotted in a histogram as shown in Fig. 6. These unique words histograms were found by iteratively determining the 15 most frequent words in each class and then adding each word that appeared in more than one class to an exclusion list. This process was repeated until no new words were added to the exclusion list; thus, all the words are unique across all classes. This gives a good representation of the most significant words for the classification of each class. This process resulted in exclusion of 85 words. These words tend to be frequently used words that have no significant relationship to one specific sentiment class. Examples of these words are ‘and’, ‘the’, ‘like’, ‘but’, ‘really’, ‘can’, ‘your’, ‘need’, and ‘know’. Intuitively, these words provide no insight us as humans as to what the sentiment of the tweet would be. Likewise, these features likely have little significance for machine learning methods. This analysis will be considered when selecting a text-to-feature method to be used in this research.

The words that represent the top 15 unique words for each class are more representative of the sentiment, and therefore useful for classification. This is not the case for all words shown in Fig. 6, however it is for the majority. Some examples of descriptive words are ‘bad’ and ‘alone’ for sadness, ‘good’ and ‘happy’ for joy, ‘anxious’ and ‘terrified’ for fear, ‘angry’ and ‘irritable’ for anger, ‘sweet’ and ‘caring’ for love, and ‘amazed’ and ‘impressed’ for surprise. These words clearly have high correlation with the sentiment class that they belong to, meaning that these words in a tweet will often be a good indication of the sentiment of that tweet. This will be considered when selecting and tuning a method to extract features, and when considering what types of features to use for training of a machine learning model to classify tweet sentiment.



(a) (b)

Fig. 5: (a) Distribution of tweet character lengths and (b) box plots for tweet lengths by class.

B. Traditional Methods

The traditional methods evaluated thus far have been primarily focused on decision tree based ensemble methods. Significant work has also been done to understand and implement different traditional feature extraction methods. In this section we will outline the progress made using decision tree ensemble methods when classifying features using bag-of-words and TF-IDF feature extraction methods. We will also present preliminary information regarding parameter tuning for models; however, more comprehensive results will be included for the final report.

1) Bag-of-Words: The first feature extraction method used was Bag-of-Words (BoW). The BoW was generated using the $sklearn\ CountVectorizer$ object. To reduce the total size of the feature space, all features with less than 5 instances in the dataset were removed. The built-in English stop word list was also used to eliminate common words such as 'the' and 'and' which do not create salient features. This resulted in a feature vector of length 3398. This is still a rather large number of features, so future work may be invested in further reduction of the feature space dimensionality.

Three different decision tree ensemble models were trained with the training and validation data and tested with the test set. The methods used were bagging, random forest, and AdaBoost. Table IIa and Table IIb show results for bagging and random forest. The performance for both methods can be observed. Both methods do show overfitting, however it is not extreme and with more tuning a better balance between the train and test accuracy of the results will likely be achieved. The accuracy of both models is nearly, the same with random forest performing slightly better overall. Both methods used 10 estimators with no limit on the maximum tree depth. Future work will consider the depth of the trees.The unbalanced nature of the dataset shows some impact on performance with the 'surprise' class having the lowest F1-Score. That class has by far the fewest samples in the dataset, which likely contributes to it's poorer overall performance.AdaBoost performance is not shown in full because using 100 estimaters proved to result in underfitting and very low performance. Additional analysis is being done to determine better parameters for the model. We believe that this poor performance may be due to too few estimators as they are 'weak' estimators as well as biasing towards the more prevalent classes. Analysis of the confusion matrix indicated it was guessing the majority class in most cases.

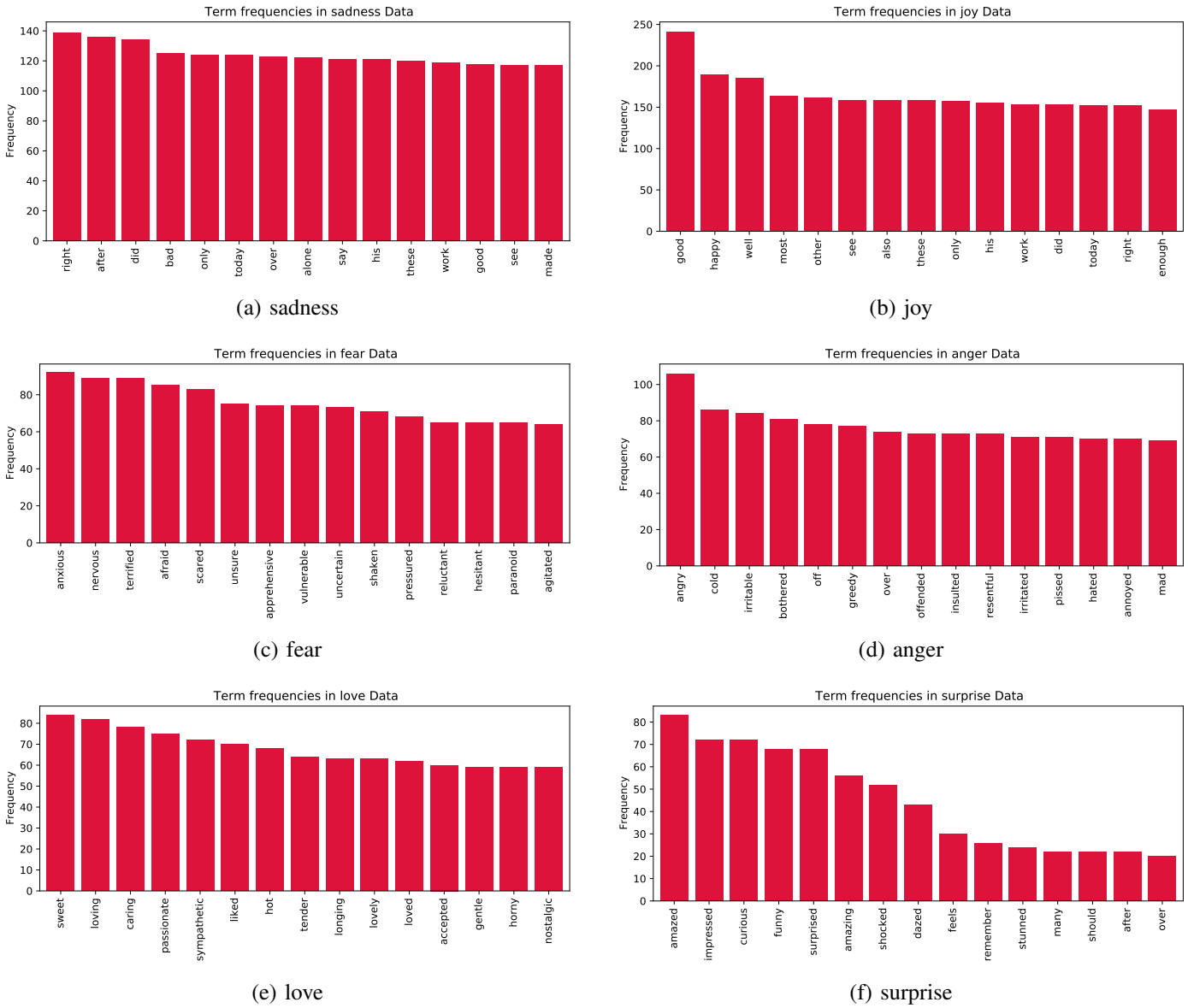
TABLE II: Bagging and Random Forest results using BoW features.

(a) Results for the Bagging with BoW features

	Precision	**Recall**	**F1-Score**	**Support**
Sadness	0.89	0.90	0.89	581
Joy	0.92	0.84	0.88	695
Love	0.71	0.75	0.73	159
Anger	0.87	0.92	0.89	275
Fear	0.83	0.85	0.84	224
Surprise	0.60	0.76	0.67	66
Accuracy			0.86	2000
Macro avg	0.80	0.84	0.82	2000
Weighted avg	0.87	0.86	0.86	2000

(b) Results for the Random Forest with BoW features

	Precision	**Recall**	**F1-Score**	**Support**
Sadness	0.92	0.92	0.92	581
Joy	0.91	0.88	0.90	695
Love	0.73	0.75	0.74	159
Anger	0.89	0.90	0.89	275
Fear	0.84	0.85	0.85	224
Surprise	0.62	0.73	0.67	66
Accuracy			0.88	2000
Macro avg	0.82	0.84	0.83	2000
Weighted avg	0.88	0.88	0.88	2000



(a) sadness (b) joy

(c) fear (d) anger

(e) love (f) surprise

Fig. 6: Top 15 unique words for each class.

2) TF-IDF: The same feature extraction and model parameters were used to train the decision tree ensemble methods using TF-IDF features. Overall performance is very similar to using BoW features. Results are shown in Table IIIa and Table IIIb. Of note is that the bagging method results for the 'surprise' class performs noticeably worse than in the other cases considered.

C. Modern Methods

1) RoBERTa: Devlin et al. [13] introduced a language model called BERT, which stands for Bidirectional Encoder Representations from Transformers. BERT is a simple, yet powerful model, having obtained state-of-the-art results on many NLP tasks. In 2019, Liu et al. [14] built on BERT with by optimizing its pretraining, creating a model called RoBERTa. We worked through a tutorial notebook based on code by Marcin Zablocki [15] that used RoBERTa to classify sentiment on another tweet dataset. The code was outdated and required significant changes to become functional again, but it eventually achieved results that exceed what we have seen using other methods. The results are shown below in Table IV.

IV. Future Plans

We plan on using the remainder of our time this semester to investigate and apply the modern methods for sentiment analysis. This includes the transformer architectures used in BERT and RoBERTa, as well as LSTM-based models that have not yet

TABLE III: Bagging and Random Forest results using TF-IDF features.

(a) Results for the Bagging with TF-IDF features

	Precision	Recall	F1-Score	Support
Sadness	0.92	0.89	0.91	581
Joy	0.89	0.90	0.90	695
Love	0.75	0.72	0.73	159
Anger	0.89	0.90	0.89	275
Fear	0.80	0.87	0.83	224
Surprise	0.61	0.58	0.59	66
Accuracy			0.87	2000
Macro avg	0.81	0.81	0.81	2000
Weighted avg	0.87	0.87	0.87	2000

(b) Results for the Random Forest with TF-IDF features

	Precision	Recall	F1-Score	Support
Sadness	0.94	0.90	0.92	581
Joy	0.89	0.91	0.90	695
Love	0.77	0.72	0.74	159
Anger	0.88	0.89	0.88	275
Fear	0.82	0.88	0.85	224
Surprise	0.69	0.64	0.66	66
Accuracy			0.88	2000
Macro avg	0.83	0.82	0.83	2000
Weighted avg	0.88	0.88	0.88	2000

TABLE IV: Results for the RoBERTa model

	Precision	Recall	F1-Score	Support
Sadness	0.966783	0.951807	0.959327	581
Joy	0.964018	0.925180	0.944200	695
Love	0.782609	0.905660	0.839650	159
Anger	0.920863	0.930909	0.925859	275
Fear	0.908654	0.843750	0.875000	224
Surprise	0.681319	0.939394	0.789809	66
Accuracy			0.923500	2000
Macro avg	0.870708	0.916117	0.888959	2000
Weighted avg	0.928936	0.923500	0.924889	2000

been explored. We would also like to continue investigating multi-class classification with all of these models. Future plans will also include additional analysis of the traditional methods presented, with particular interest placed on parameter selection.

Overall, our progress to this point has given us a strong fundamental knowledge of the techniques used for sentiment analysis classification. In our remaining time working on this project, we would like to build upon this base, understanding the nuances and tradeoffs of specific models used for this task. Our final report will provided detailed analysis of these tradeoffs along with comparison of different machine learning methods for sentiment analysis.

REFERENCES

[1] R. Waykole and A. Thakare, "A review of feature extraction methods for text classification," *International Journal of Advance Engineering and Research Development*, vol. 5, 2018.

[2] E. Saravia, H.-C. T. Liu, Y.-H. Huang, J. Wu, and Y.-S. Chen, "CARER: Contextualized affect representations for emotion recognition," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 3687–3697. [Online]. Available: https://www.aclweb.org/anthology/D18-1404

[3] J. Deriu, A. Lucchi, V. D. Luca, A. Severyn, S. Müller, M. Cieliebak, T. Hofmann, and M. Jaggi, "Leveraging large amounts of weakly supervised data for multi-language sentiment classification," 2017.

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

[5] M. Rathi, A. Malik, D. Varshney, R. Sharma, and S. Mendiratta, "Sentiment analysis of tweets using machine learning approach," in *2018 Eleventh international conference on contemporary computing (IC3)*. IEEE, 2018, pp. 1–3.

[6] A. Go, R. Bhayani, and L. Huang, "Twitter sentiment classification using distant supervision," *Processing*, vol. 150, 01 2009.

[7] P. M. Sosa, "Twitter sentiment analysis using combined lstm-cnn models," *Eprint Arxiv*, pp. 1–9, 2017.

[8] U. of Michigan, "Umich si650 - sentiment classification dataset," 2011. [Online]. Available: https://www.kaggle.com/c/si650winter11

[9] N. Sanders, "Twitter sentiment corpus," 2011. [Online]. Available: https://github.com/zfz/twitter_corpus

[10] N. Chen and P. Wang, "Advanced combined lstm-cnn model for twitter sentiment analysis," in *2018 5th IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS)*, 2018, pp. 684–687.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[12] T. P. Galassi A., Lipp M., "Attention in natural language processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, 2021.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[14] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019.

[15] M. Zabłocki, "Custom classifier on top of bert-like language model - guide," Mar 2020. [Online]. Available: https://zablo.net/blog/post/custom-classifier-on-bert-model-guide-polemo2-sentiment-analysis/