Real or Not? NLP with Disaster Tweets

Kristopher Flint and Zachary Kellerman

Goal of Our Project

As the field of machine learning progresses we are beginning to be able to train computers to analyze the complex structure of the English Language. This is becoming increasingly important as humans can no longer reasonably examine the user content produced by social media.

In this research, we use natural language processing to predict whether or not text is referring to a natural disaster. Using the Kaggle competition data, consisting of 10,000 labeled tweets, we apply machine learning models to determine which tweets refer to real disasters and which ones do not.

Exploratory Data Analysis

When we began our research on this data, we started with investigating the structure of it. One thing we noticed very quickly was that there were a lot of patterns to differentiate the two types of tweets. Many tweets classified as a disaster contained words indicating some type of serious accident, weather, or criminal activity such as “crash”, “storm”, and “killed.” The non-disaster tweets contained words and phrases such as “news”, “youtube video”, and “people.” We cannot go off word frequency alone though, so we applied TF-IDF to the two different types of tweets and noticed the exact same trends. We also decided to generate a couple of word clouds to get a better view of the word frequencies in our data.

These word clouds represent the frequencies of the words for disasters(left) and non-disasters(right) in our dataset. They give us a view of what our dataset contains.

Preprocessing And TF-IDF Training

For our first models (naive bayes, SVM, random forest, ensemble learner, and sequential neural network) we used the nltk library to do our preprocessing. Using this library we were able to make our text lowercase for uniformity and created a token for each unique word. We then separated the tokens into dictionaries based on what kind of word they are (noun, adjective, verb, and adverb). With these dictionaries we then applied a lemmatizer which tries to remove endings from a word like ly, ing, s, etc. while trying to preserve the root meaning of the word. This lemmatizer allows us to be able to compare words like volcano and volcanoes. Now that we have this preprocessed list of tokens we can run our TF-IDF function on it. TF-IDF assigns number values to every token in the list based on their frequency in the dataset. These numbers are all between 0 and 1 and allow us to numerically represent the text in order to run it through our prediction models.

For our Bert model, we were able to use a more sophisticated preprocessing algorithm, in the Bert library, that had already been pre trained for English on the Wikipedia and BooksCorpus. This allows us to achieve a higher accuracy with a quicker notebook. The preprocessing model creates three lists for our final neural network. These lists include the tokens, masks, and segment ids. The tokens are the actual words themselves all padded to be the same length, as required by neural networks. The masks tell us which part of the tokens are the characters and not just whitespace. The segment ids allow us to keep track of which tweet each word is associated with, each tweet is given a unique segment id.

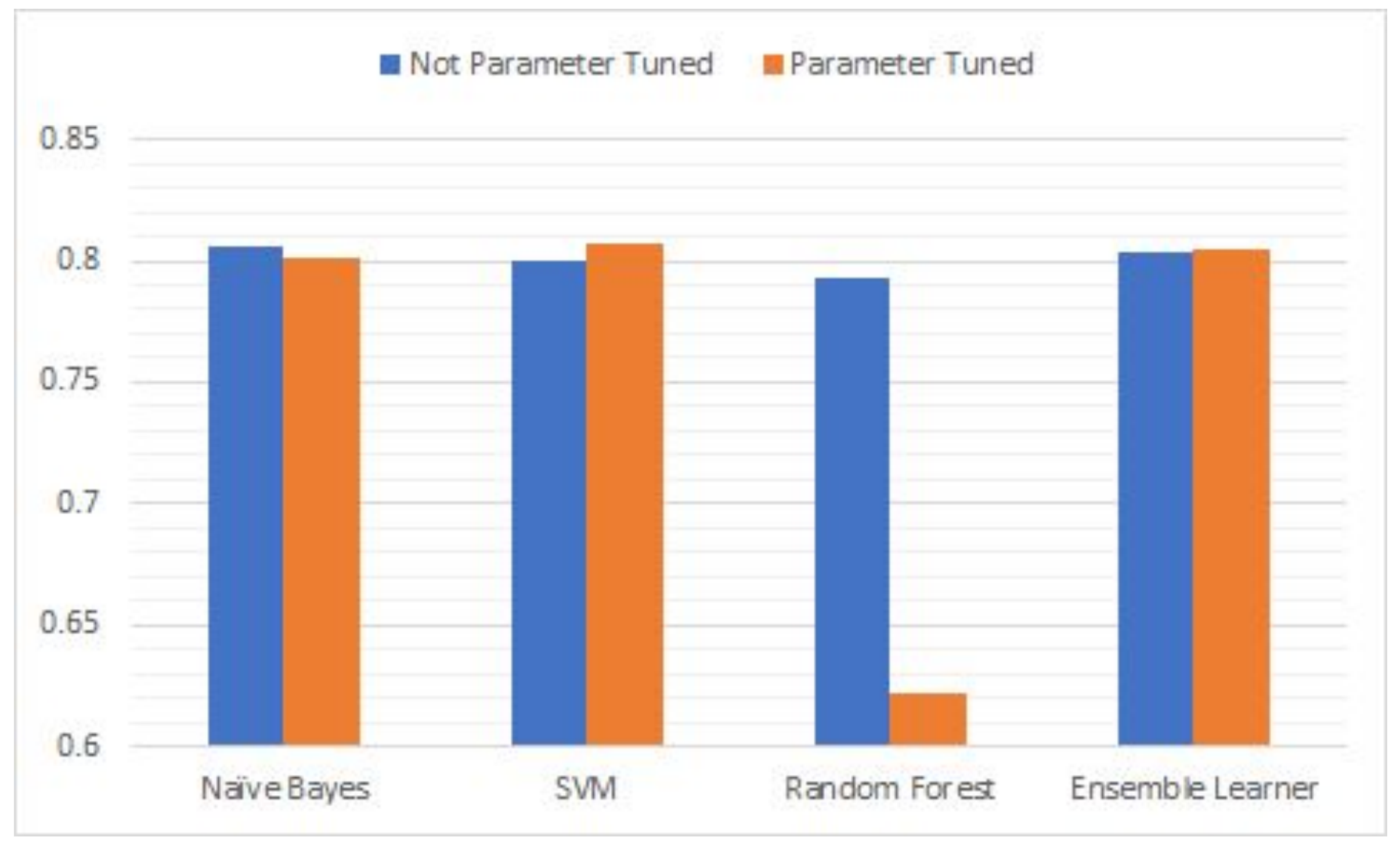
Now that we have our preprocessed text we can move on to our prediction models.

Model Selection

- Naive Bayes: We chose to go with the multinomial naive bayes classifier because it is one of the best models to use for this type of text classification.
- Support Vector Machines (SVM): We decided that an SVM classifier would do well for our research because it tries to find the best line that will accurately split the data up in one of the two categories.
- Random Forest: We chose a random forest classifier because of how it uses its decision tree process to decide if a tweet is referring to a disaster or not.
- Ensemble Learner: We used a majority rules voting classifier that used the previous three models as its estimators.
- Sequential Neural Network: We chose a sequential neural network that uses Word2Vec because with the right word vectorizer, this can be a powerful model.
- BERT: We chose to create a BERT model because it is Google’s NLP framework and we wanted to implement it into our research to see how well it would do.

Results

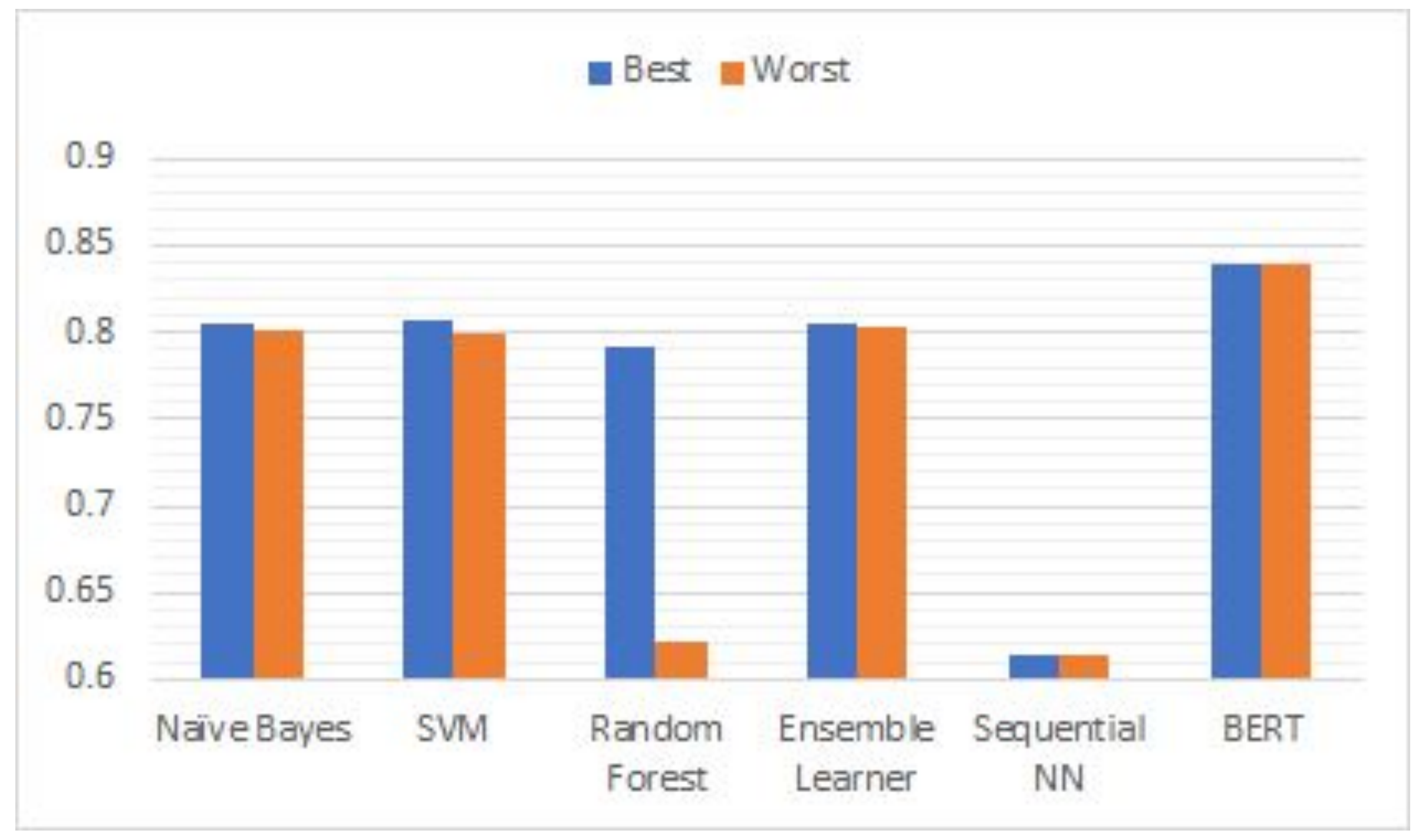
For the first four models indicated above, we had an accuracy percentage that averaged at about 80%. Our sequential neural network model had an accuracy of about 61.5% and we feel that this is due to our word vectorizer not being optimized as well as it could be. After these submissions we decided to tune our parameters and implement cross validation. Even after tuning our parameters, we saw the accuracy of the Naive Bayes, SVM, and the Ensemble Learner at about 81%. The parameter tuned Random Forest model only achieved 62% accuracy and is most likely due to not choosing the best values for the parameters of that model.. The BERT model was the best model created with an accuracy of approximately 84% to determine if a tweet was talking about a disaster or not.



This graph compares the non-parameter tuned models and parameter tuned models which indicates how much parameter tuning helped our harmed our models.

Future Work

There are many areas in which these models can be improved upon to obtain a better accuracy at classifying tweets. For instance, we feel that one area of improvement would be to create a better Word2Vec word vectorizer and apply it to most of our models. By optimizing the word vectorizer, we could see a significant increase in the accuracy of our models. Another area that could use some additional work is parameter tuning. Overall there was not much of a difference in the accuracy of our models when parameter tuning with cross validation was applied. The only major change was to the Random Forest model and it went in the opposite direction we wanted to see. We believe that with the right parameters and right values for those parameters, there could be more positive change rather than negative. Lastly, we believe that it would be possible to see a significant improvement if we performed the models on the bigrams and trigrams that are able to be generated from the dataset. This would allow our models to get a better idea on how the words in the tweet are being used. Our work only took unigrams into account so it is very likely that we could see higher accuracy by introducing bigrams and trigrams as features in our data.



This graph compares the best accuracy and the worst accuracy we achieved for our models. This is based upon submissions with and without parameter tuning. Note: there was no parameter tuning for sequential neural net and BERT models.

Acknowledgments to Tennessee Technological University, Dr. William Eberle, and Kaggle