**CS 562 Project Report**

**Firearms Support Classification**

**Abhi Vora, Devosh Mathivanan, Rohit Mathew**

**Abstract**

This project revolves around building a classifier from scratch. We used manually collected and labelled Twitter data to help us build a classifier that accomplishes a very specific goal: to figure out if a Tweet is pro- or anti- gun. This report presents the architecture and results of a number of models tried, along with possible improvements and uses for the final model arrived at.

**Introduction**

In light of recent events that range from the United States presidential election to school shootings, the U.S. has seen a massive upheaval of opinions in terms of firearms and their place outside an armed forces setting. This includes making people’s opinions heard on social media. We decided to harness this stream of data to try and understand if there has been a significant shift in the way people view firearms, or if they remain true to their roots, unfazed by the ethical dilemma and the political turmoil around them.

We started this project with a specific goal in mind: to collect Tweets that, in some way, spoke about gun- related topics. We then feed these Tweets into a black box algorithm that tells us whether a Tweet is in support of stricter gun laws (or just generally against guns) or against the idea of limiting the peoples’ right to carry firearms (for guns). We then hoped to run some sort of clustering algorithm (like *k*- means) on these outputs to help us view, at varying granularities, people’s opinions on guns.

We soon figured out that the black box algorithm would prove to be the most difficult part of this project, and this stage would take up most of our resources. This project went from “Figuring out America’s Opinions on Guns” to “Building an Opinions on Guns Classifier”. We present the various models we tried, along with the numerous combinations of parameters we used, their effectiveness, and ways to improve the results, along with an application and its’ result of the model we finally arrived at.

**Problem Definition**

Given the text of a Tweet that we know in some way talks about guns, we try to figure out if this text talks about firearms in a positive or a negative way. Formally, we hope to classify every Tweet into one of three classes: “For guns?”, “Against guns?”, or “Doesn’t make sense”. The third label is assigned for those Tweets that, independently, don’t contribute a valid reasoning to the debate (e.g.: lone emoticons, or acronyms).

To achieve this, we first need to collect suitable Tweets, clean them, label a subset of them, transform them, experiment with different classifiers, and compare accuracies. Below, we list the various classifiers we worked with, as well as the parameters we tweaked to help us achieve our results.

* **Bag- of- Words Model:**

A classic text pre-processing technique where every Tweet text is represented as a fixed- length feature vector. Every location of the vector gives us the count of a particular word associated with the location for that Tweet. This representation does not care for word order.

* **Term Frequency- Inverse Document Frequency:**

Another pre- processing technique that is a close relative of the Bag- of- Words model. TF- IDF is essentially a weighted version of the model, where more important, less common words are given higher weights, while less important, more common words are given a lower weight.

Term Frequency(w) = Number of times w appears in a Tweet/ Total number of words in Tweet

Inverse Document Frequency(w) = log(Total number of Tweets)/ Number of Tweets with w in it

So TF- IDF(w) = Term Frequency(w) \* Inverse Document Frequency(w)

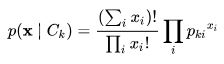
These weights (TF- IDF(w)) help map the Tweet text into vector space.

* **Word Embeddings:**

This is a text pre- processing technique born out of Deep Learning. Here, words themselves are individually mapped into a vector space, and these mappings are used to form sentences that represent a Tweet. The beauty of this representation is that conceptually similar words are mapped close to each other in the vector space. This works especially well when building certain Deep Learning models.

* **Naïve Bayes Classifier:**

This is a supervised probabilistic model that performs a classification task. More specifically, we used the Multinomial version of the Naïve Bayes model, where the Bag- of- Words and TF- IDF representations were used as feature vectors x and class labels (binary and ternary) were defined with C:



* **Support Vector Machine:**

Another supervised probabilistic classification model that specializes in binary classification. Having three classes, we worked with splitting our data so we could to build different models for each pair of classes, similar to a one- vs- all method of multiclass classification.

* **Random Forests:**

A supervised ensemble learning model, built off of decision trees, which works with classification. Here too we experimented with different models for each pair of classes as well as a ternary model for multiclass classification.

* **Long Short- Term Memory:**

A unit used in Recurrent Neural Networks. LSTM networks work particularly well with sequences of data, in this case, sequences of word embeddings, or sentences. This is thanks to its very architecture, which includes *gates* that help it with selectively remembering and forgetting features.

**Approach**

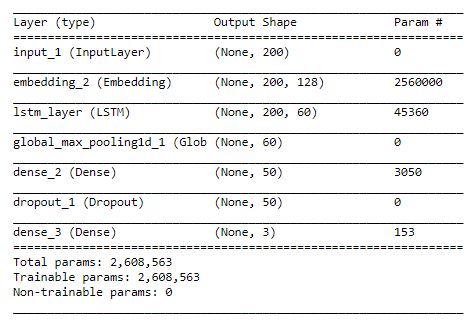
We started out with using the Twitter API to collect real- time Tweets. This was tricky, because the API supports filtering the Tweets on any one parameter, while we wanted to filter Tweets based on two parameters: location (limited to the U.S.) as well as keywords (containing words like “guns”, “Second Amendment”, “shooting”, etc.). We also made sure, during streaming, that the “location” field of a Tweet was in a specific format, since we knew that we are going to work with this feature at some point of time, and this extra step during Data Collection would prove to be favorable later on. These Tweets were saved in a NoSQL setting, namely MongoDB, which was perfect for working with JSON formatted data. Over the period of many weeks, we collected about 15,000 data points in the specific format as mentioned.

The next step was pre- processing the “text” part of the Tweets. We applied a number of cleaning techniques to these data points, which range from stripping out new- line characters, retweeted data, and ‘@’ values, and then passing them on to transformation algorithms that convert texts into the vector space representations mentioned above. We preserved certain features through these transformations that we would use later on in the project: an identifier for each Tweet, state from which the Tweet was sent.

We also took the time to manually label ~500 Tweets to train and test our classifiers. Since this was a multiclass classification problem, we had three labels: “For guns?”, “Against guns?”, and “Makes no sense”, the implications of which are mentioned above.

With these feature vectors, we worked towards building our models. Something to note is that when examining our dataset, we noticed ~44% of our Tweets belonged to the class that didn’t make sense, while 37% belonged to Tweets against guns. In order to combat this imbalance and reduce the risk of overfitting while building our models, we ensured that we oversampled the Tweets for guns, and the Tweets that didn’t make sense, when building the respective binary classifiers.

We first started out with an LSTM network, naively assuming that Deep Neural Networks would give us the best accuracy. We used the Word Embedding representation of the Tweets, with each Tweet represented as a vector of length 200. Since we have three classes, we built 2 different versions to help us out: the first version was a (ternary) multiclass classifier, and the second version had two phases: first to distinguish between Tweets that didn’t make sense and everything else, then to distinguish between Tweets that supported guns and those that didn’t. However, neither of these approaches did any better than guessing the outputs: for the first model, we achieved 66% accuracy with unsampled data, and 33% accuracy with sampled data. For the second model, we achieved 50% for both binary classifiers with sampled data. On examining the predictions, it was noticed that the model predicted every Tweet to be “Against Guns” thus giving us the above accuracies. We tried to combat this overfitting further by introducing Dropout layers, which act as regularizers, but that didn’t help. Given below is the summary of one of the final LSTM models we used for multiclass classification. Note that the final activation function with three outputs is a sigmoid function:



We learnt that while Neural Networks are, in fact, quite effective for a task like this, they rely on large amounts of data to work accurately. Working with 500 data points does not count, so we decided to stop looking for answers in this area.

We moved on to classical models to build our classifier.

We used Multinomial Naïve Bayes with Bag of Words and TF- IDF representations of data. We also trained them on pairwise labels, adopting the one- vs- all training method for the two phases described above. We found this model to work quite effectively with the Phase 2 Bag- of Words representation.

We also tried an SVM model on the second phase (for/ against guns) and this gave us an accuracy that stayed at around 72%. Please note that this was on unsampled (unbalanced) data.

The Random Forest model gave us satisfactory accuracy. It gave us >75% accuracy for the Phase 1 binary classifier “For guns” vs “Against guns” but less than 50% for the Phase 2 binary classifier “Doesn’t make sense” vs everything else. We decided to compromise and use this model for a multiclass classifier with three labels; this gave us an accuracy of 64%, much better than anything we tried so far. Note that we use the Bag- of- Words representation since this consistently gave us better results for the purposes of this project. This is the final model we decided to use.

**Results**

Here are the results of all the models described above, for the combinations of different pre- processing techniques, models and parameters for the multiclass classification and for the different phases of binary classification.

**Binary Classification:**

* **Phase 1**

|  |  |  |
| --- | --- | --- |
| Model | Pre- processing | accuracy |
| Random forest | Bag- of- Words | < 50% |
| LSTM Network | Word Embedding | 50% |

* **Phase 2**

And with the Neural Network:

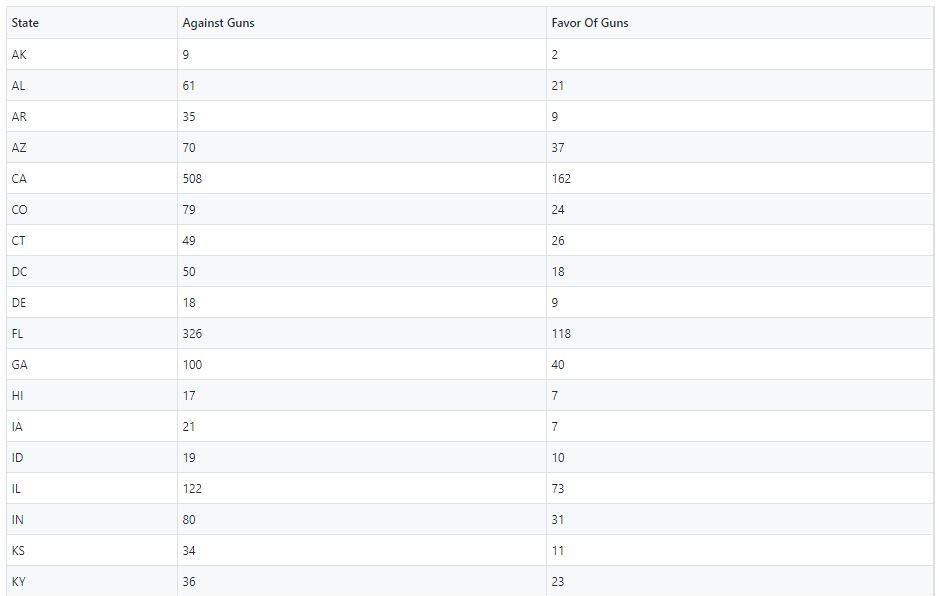
|  |  |  |
| --- | --- | --- |
| Model | Pre- processing | accuracy |
| LSTM Network | Word Embedding | 66% |

**Multiclass Classification:**

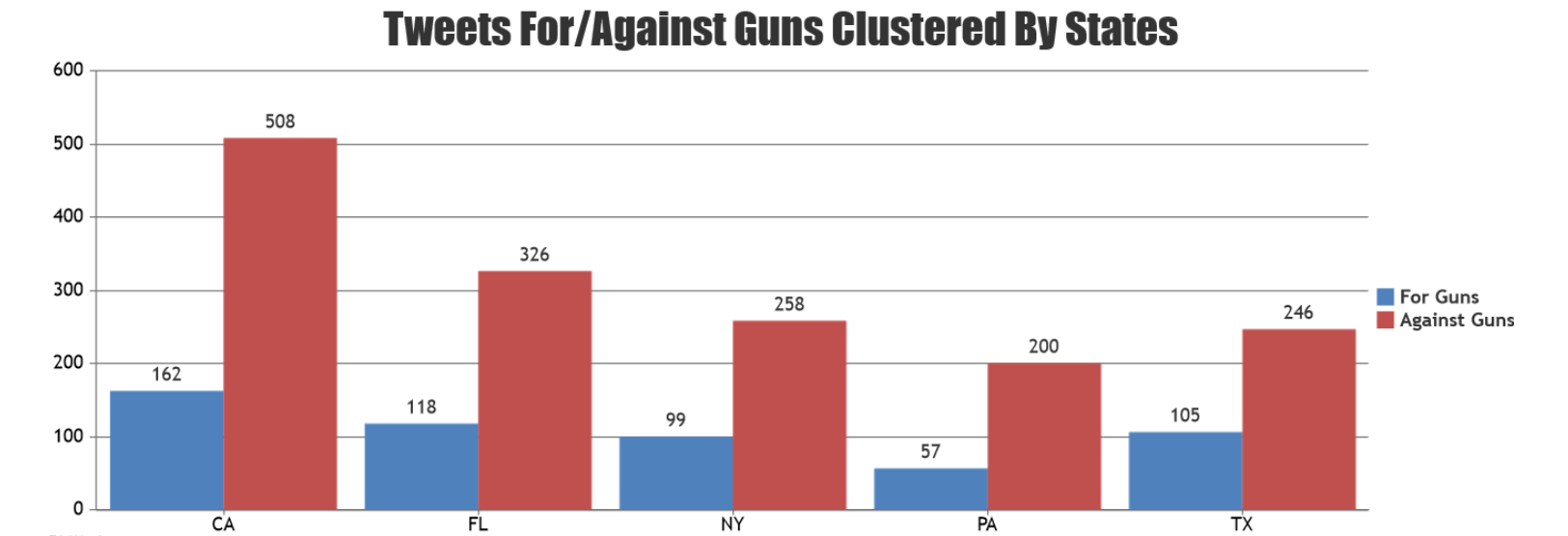
|  |  |  |
| --- | --- | --- |
| Model | Pre- processing | accuracy |
| Random forest | Bag- of- Words | 68% |
| LSTM Network | Word Embedding | 66% |

Please note that the only reason the LSTM network gave us 66% accuracy in all cases is because it labelled ALL the tweets as “Against guns” for unsampled (unbalanced) datasets. As mentioned, we tried to fix this overfitting problem in many ways, but nothing worked.

Using the multiclass Random Forest model that we were satisfied with, we went ahead and tried a real- world application of our model. We classified all the data that we collected into the three known labels, and using SQL we grouped the results by state, getting rid of the Tweets that belonged to the label “Doesn’t make sense”, leaving behind Tweets that were on either end of the spectrum. Given below are the final results from some of the states:



Here are the distributions from some of the states that we found to be interesting:



We notice several thought- provoking features:

* Every state has a higher number of Tweets against guns (almost 75% of Tweets against guns)
* Since most of the data was collected in the aftermath of the Florida school shooting and the YouTube headquarters shooting, it would make sense that the majority of people posting about this topic on social media would rally against guns, while a small number tries to defend their point of view
* Texas, one of the strongest Republican and pro- gun states, had the smallest difference of the 5 (almost 30% of the Tweets supporting gun ownership).

**Conclusion**

The model that we have built performs quite well, given the circumstances and limited amount of data that we had to work with. This model can be used for a number of projects, including the idea we first began with: to examine how people’s opinions toward firearms have changed ever since Donald Trump became president, and following any sort of event that warrants talks about guns.

The accuracy of our models can be improved, by using Deep Neural Networks provided we large amounts of data, by training multiple statistical classifiers and using ensemble methods, by using certain heuristics along with the actual Tweet to perform classification, and by playing around with the various parameters available to us. We can explore these methods more confidently, given the strong foundation we now have with working with Tweets, classification models, and various Natural Language Processing techniques.