Improving Incident Response of the American Red Cross in the Greater Chicago Area by Using Text Classification of Posts From Twitter

Project Members:
Manager: Pavan Sistla
Scribe/Researcher: Evan De Broux
Coder: David Cho
Coder: Christopher Millan
Coder: Henry Post
Researcher/Coder: Trung Pham
Researcher: Raul Renteria
Researcher: Hasani Valdez
Researcher: Rena Haswah

April 2019

Contents

1 Abstract

Tweets on Twitter about true disasters are hard to easily detect and are often hidden among a sea of irrelevant tweets about only tangentially related content. We introduce a suite of tools to obtain tweets and attempt to classify them as relevant or irrelevant with regards to house fires and other disasters using a variety of machine learning models.

2 Introduction

The American Red Cross (ARC) is an organization has provided emergency assistance, disaster relief, and disaster preparedness and education across the United States since its founding in 1881. On average, the American Red Cross responds to more than 62,000 disasters a year, which is around one disaster every 8 minutes, 90 percent of which are home fires. The truly remarkable fact is that of their entire workforce that goes out of their way to help others in need, 95 percent of their workers are volunteers. In the Northern Illinois region, the ARC has received over 6,000 calls in the past 5 years and has provided humanitarian aide to approximately 75 percent of those incidents.

Past IPRO groups who have worked for the ARC through their representative, Jim McGowan, and our instructors, Bo Rodda and Matt Robison, have given them useful insights into the data that the ARC has collected. The summer of 2018 IPRO group has shown that the fires in the City of Chicago are spatially correlated with each other. The fall of 2018 IPRO group built a fairly accurate model in predicting fires at the neighborhood level using demographic data from the Census Bureau and the City of Chicago Data Portal. This gave the ARC useful insights to answer questions such as, "which neighborhoods should we focus our education efforts in?" and "which groups of people should we be targeting to better prepare them for fires?"

Despite some early exploration into the past project groups' models, there was a different point that piqued our research groups' interest. Mr. McGowan mentioned that a majority of the incidents that the ARC learns about come from social media, with specific emphasis on Twitter. During a visit on March 5th, 2019, Mr. McGowan noted that around 44 percent of the ARC's reported incidents come from Twitter. This led us to ask multiple questions. How does the ARC identify reliable sources on these social media platforms?

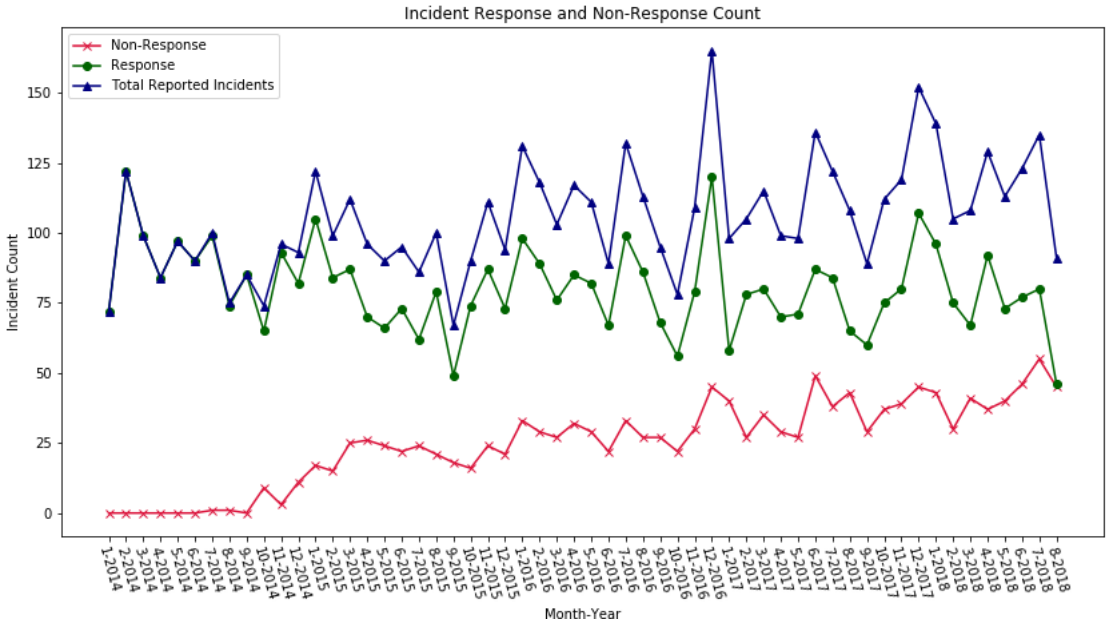

Figure 1: The running count of incidents the ARC has in their records up until August 2018. We notice that despite some fluctuations in number of reported responses by month, they stay fairly steady. However, the number of incidents that the ARC did not respond to has gradually increased over this time period and additionally the ARC is unaware of many incidents in the Chicago area. Our goal is to have the ARC to be aware of more of these incidents.

Is it possible to evaluate the reliability of their social media sources? Can we identify new sources that report incidents, or find where fires are occurring before they are reported on more official sources? After a brainstorming session, our group came up with a potentially useful tool: a program that goes through geo-tagged tweets on Twitter and identifies those that are useful to the disaster response efforts of the ARC.

3 Text Classification Tool

In creating this text classification tool, there are 2 (maybe more) processes that we needed to create and fine-tune: the process of "scraping" the posts from Twitter and the process of classifying the tweets.

3.1 Twitter Scraper

The Twitter Scraper aspect of our project is a tool that simply exists to gather massive amounts of tweets with minimal effort. It exists as a Python package on the Python Package Index site, and is easily to install. Currently, it supports a handful of different methods to gather tweets.

3.1.1 Scraping by keywords

The first method that it can use to gather tweets is by keywords that the tweets contain. The scraper can take a single keyword and a count of results, or many keywords. Below is code showing the usage of the keyword-scraping feature:

```
# Returns a total of 600 tweets
results = scraper.scrape_terms(
    terms={"fire", "#housefire", "firedamage"},
    count=200
)
```

Listing 1: Scraping multiple keywords

This snippet of code will return 200 tweets each about the term "fire", the hashtag "#housefire", and the term "firedamage" for a total of 600 tweets.

3.1.2 Scraping by keywords restricted by location

The next feature our scraper has is one which limits the areas that the scraper operates in. This is an augmentation of the scraper's ability to search for keywords and hashtags, and acts as a filter on that function.

This is important because most of the work that the ARC of Chicago wishes to do takes place in or around Chicago.

Below is a code example of the same search, but restricted to within 50 miles of Chicago's center.

```
# Returns 600 tweets that occur 50 miles from Chicago
results = scraper.scrape_terms(
    geocode="41.8297855,-87.666775,50mi",
    terms={"fire", "#housefire", "firedamage"},
    count=200
)
```

Listing 2: Geo-location scraping filter

This time, we still get 200 tweets per term, but only ones that have been tagged within Chicago.

Instead of searching all of twitter for the three terms, only tweets that individuals have chosen to tag with a geographical location within a 50 mile radius of latitude 41.8, longitude -87.6, which is Chicago's center.

3.1.3 Scraping by account

This feature allows one to scrape the most recently tweeted tweets by one or more accounts.

This is to make it easy to repeatedly query accounts, or to get large amounts of tweets from a set of accounts.

Below is a code example of the scraper getting the top 100 tweets from various Twitter accounts that the RedCross account follows.

```
# Returns 300 tweets from 3 red cross accounts, 100 each.
results = scraper.scrape_accounts(
    accounts={"@RedCross", "@NWSChicago", "@MABASIllinois"},
    count=100
)
```

Listing 3: Scraping by account

The syntax is very similar to that of the previous scraping methods. This snippet of code would result in a total of 300 tweets, 100 from each Twitter account.

3.1.4 Saving to a CSV file

This feature allows one to save scraped tweets to a comma-separated value file, to perhaps later be analyzed in a spreadsheet program like Microsoft Excel or be loaded into another data analysis program.

```
# Returns 300 tweets from 3 red cross accounts, 100 each.
results = scraper.scrape_accounts(
    accounts={"@RedCross", "@NWSChicago", "@MABASIllinois"},
    count=100
)

# Saves above results to a CSV file called 'output.txt'
scraper.save_statusdict_to_csv(results, 'redcross_tweets.csv')
```

Listing 4: Saving to CSV

3.1.5 Other features

The scraper allows you to scrape many more tweets than twitter normally allows you at any one time, and will wait when it hits the Twitter API rate limit to finish scraping tweets.

3.2 Proposed Classification Models Tested

The classifier that was built in this semester's IPRO was fairly basic. The classifier is a word-to-vector machine that tries to measure the dependence of one unigram or bigram in relation to other unigrams or bigrams, and tries to use those measurements to place a tweet in a classification. The research group started by testing out different text classification models: a Naive Bayes Classifier, a Linear Support Vector Classifier, a Logistic Classifier, and a Random Forest classifier. There were no attempts to tune hyper-parameters used for these classifiers, no additional tf-idf smoothing, no testing for using additional leafs for the random forest classifier, among other features that could have been tested due to time constraints. Working on "fine-tuning" these parameters to generate better classifications for each model could be taken up as future work. We discuss the basic mathematics behind the Naive Bayes, LinearSVC, and Logistic classifiers below.

3.2.1 Naive Bayes Classifier

One of the text classification model will rely on a Naive Bayesian classifier in order to classify the tweets that we pull through Twitter's API. The tweets can be assigned a label from a finite set based on the characteristics of the tweet. The key is that the classifier will assume independence of the characteristics of the tweet, independence of the words used to construct the tweets, that would be used to determine the probability that the tweet should be assigned one of the classes.

Mathematically, the probability that a tweet would belong to a certain class, like "fire", is conditional. If we assume the features of the tweet are independent, have a tweet with n features that are relevant to the classification of the tweet, this can be stored as a vector, $x = (x_1, x_2, ..., x_n)$. Say that there are k different classes, $\{C_1, C_2, ..., C_k\}$, which the tweet could potentially be associated with. Based on the features of the tweet, we can assign a conditional probability to which class the tweet could belong to for any class $C_i, i \in \{1, 2, ...k\}$:

$$P(C_i|x_1, x_2, ..., x_n) \tag{1}$$

The obvious problem with this line of thinking is that the potential number of features for a tweet can be relatively large. Using Bayes' Theorem, the probability a tweet being associated with class C_i, would be:

$$P(C_i|x) = \frac{p(C_i)p(x|C_i)}{p(x)} \tag{2}$$

In other words, the probability that a tweet belongs to class i is equal to the prior likelihood that a tweet belonged to class i times the likelihood probability that this tweet belongs to class i, divided by the likelihood that the specific tweet exists in that order. Note that the numerator is really the only thing of interest here, as it is dependent on class k. Using the laws of conditional probability, we can rewrite the joint probability of the model:

$$P(C_i, x_1, ..., x_n) = P(x_1|x_2, ..., x_n, C_k)P(x_2, ..., x_n, C_k) \tag{3}$$

$$= P(x_1|x_2, ..., x_n, C_k)P(x_2|x_3, ..., x_n, C_k)P(x_3, ..., x_n, C_k) = ... \tag{4}$$

$$= P(x_1|x_2, ..., x_n, C_k)P(x_2|x_3, ..., x_n, C_k)...P(x_n|C_k)P(C_k) \tag{5}$$

Yet, naive conditional independence implies that for any feature x_j is conditionally independent of x_l, for any $j \neq l$, which translates to:

$$P(x_j|x_{j+1}, ..., x_n, C_k) = P(x_j|C_k) \tag{6}$$

This changes the conditional probability to:

$$P(C_k|x_1, ..., x_n) \propto P(C_k, x_1, ..., x_n) \tag{7}$$

$$= P(C_k) \prod_{i=1}^{n} P(x_i|C_k) \tag{8}$$

Using the independence assumptions, the probability of observing class C_k, given features $\{x_1, ..., x_n\}$ is:

$$\frac{P(C_k) \prod_{i=1}^{n} P(x_i|C_k)}{P(x_1, ..., x_n)} \tag{9}$$

$$= \frac{P(C_k) \prod_{i=1}^{n} P(x_i|C_k)}{\sum_k P(C_k) P(x_1, x_2, ..., x_n|C_k)} \tag{10}$$

This is how probabilities are solved for using any distribution to determine $P(C_k|x_1, ..., x_n)$. We are using the multinomial distribution to determine the class probabilities. This means that $P(x_1, ..., x_n|C_k)$ is written as:

$$P(x_1, ..., x_n|C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i p_{ki}^{x_i} \tag{11}$$

where p_{ki} is the probability of class k associated with observation i.

3.2.2 Logistic Model

Logistic regression model classification is fairly simple. Define a single logistic response variable to be y_i, which is an ordinary Bernoulli random variable, i.e. a binary response variable that can take on values of 0 or 1, with:

$$P(y_i = 1) = \pi_i P(y_i = 0) = 1 - \pi_i \tag{12}$$

which means y_i has a probability distribution of:

$$f_i(y_i) = \pi_i^{y_i} (1 - \pi_i)^{1-y_i}, \forall i = 1, ..., n \tag{13}$$

which after compiling all y_i for $i = 1, 2, ..., n$ assuming independence between observations y_i has joint probability distribution:

$$g(y_1, ..., y_n) = \prod_{i=1}^{n} f_i(y_i) = \prod_{i=1}^{n} \pi_i^{y_i} (1 - \pi_i)^{1-y_i} \tag{14}$$

To find the maximum likelihood estimates of the probability we can utilize the logarithmic transformation:

$$ln(g(y_1, ..., y_n)) = \sum_{i=1}^{n} y_i ln(\frac{\pi_i}{1-\pi_i}) + \sum_{i=1}^{n} ln(1-\pi_i) \tag{15}$$

which yields:

$$1 - \pi_i = \frac{1}{1 + e^{\beta_0 + \beta_1 x_{i1} + ... + \beta_{n-1} x_{i,n-1}}} \tag{16}$$

$$\rightarrow ln(\frac{\pi_i}{1-\pi_i}) = \beta_0 + \beta_1 x_{i1} + ... + \beta_{i,n-1} x_{i,n-1} \tag{17}$$

This quantity on the right outputs a probability that y_i occurs given the observed independent variables $x_{i1}, ..., x_{i,n-1}, \forall i \in \{1, ..., n\}$. [1]

However, the case above does not involve multiple responses, or polytomous logistic regression for nominal responses, but a single response. This is necessary as we have multiple categories, but it also complicates the mathematics. Say we have J different responses categories, where $y_{ij} \in \{0, 1\}$, which is the case where case i has response j. We assume that only one category can be selected for response:

$$\sum_{j=1}^{J} y_{ij} = 1 \tag{18}$$

We let the probability that case i is in category j be represented by $P(y_{ij} = 1) = \pi_{ij}$. For J polytomous categories, there would be $J(J-1)/2$ pairs of categories and thus the same amount of linear predictors of form:

$$\pi'_{ij'j} = ln(\frac{\pi_{ij'}}{\pi_{ij}}) = X'_i \beta_{j'j}, \forall j' < j; j', j \in \{1, ..., J\} \tag{19}$$

If we assume a baseline, like the "NO" category that classifies tweets that are irrelevant to our search, then we can treat this as class J and we know that the logits for the J^{th} comparison are:

$$\pi'_{ijJ} = ln(\frac{\pi_{ij}}{\pi_{iJ}}) = X'_i \beta_{jJ}, \forall j \in \{1, 2, ..., J-1\} \tag{20}$$

Since the comparisons are being made against this baseline, we assume these probabilities are now:

$$\pi'_{ij} = ln(\frac{\pi_{ij}}{\pi_{iJ}}) = X'_i \beta_j, \forall j \in \{1, ..., J-1\} \tag{21}$$

We can always find the difference in any two responses from these $J-1$ logits as for any $k \neq l$:

$$ln(\frac{\pi_{ik}}{\pi il}) = ln(\frac{\pi_{ik}}{\pi_{iJ}} * \frac{\pi_{iJ}}{\pi_{il}}) = X_i'\beta_k - X_i'\beta_l \tag{22}$$

This yields the $J-1$ logit expressions for the category probabilities in case i:

$$\pi_{ij} = \frac{e^{X_i'\beta_j}}{1 + \sum_{k=1}^{J-1} e^{X_i'\beta_k}}, \forall j \in \{1, ..., J-1\} \tag{23}$$

[1]

3.2.3 Linear Support Vector Classification

We will define how a linear support vector machine works mathematically in a simple case. Say we have a set of training data $(x_i, y_i), \forall i \in \{1, ..., n\}$. The only values that y_i can take on are 1 or -1 which classifies the data x_i (the support vectors) into one of two classes, where x_i has dimension p. There will be some "maximum margin hyperplane" such that we can divide the data into the two classes for which the distance from the hyperplane to the nearest vector x_i is maximized. It is possible to write any hyperplane for a set of points x as:

$$w \cdot x - b = 0 \tag{24}$$

where w and x are vectors of the same dimension and b is a real number. w is the normal vector to the hyperplane. Let us assume that the data that we have been given is linearly separable. Given this, the data could be standardized to satisfy the equations one of two equations:

$$w \cdot x - b = 1, \tag{25}$$

or:

$$w \cdot x - b = -1 \tag{26}$$

depending on the classification of the data. The data also cannot fall into this "hard margin", so we need to add two constraints on the data:

$$w \cdot x - b \geq 1, given : y_i = 1 \tag{27}$$

$$w \cdot x - b \leq -1, given : y_i = -1 \tag{28}$$

This can be turned into a linear optimization problem:

$$\begin{aligned} \text{minimize: } & ||w|| \\ & \text{subject to: } y_i(w \cdot x - b) \geq 1, i = 1, ..., n \\ & y_i \in \{0, 1\}, j = 1, ..., n \end{aligned}$$

2

If the data is not linearly separable, then a "soft margin", then we have a loss function which tries to compensate for this lack of separation:

$$max\{0, 1 - y_i(w_i \cdot x_i - b)\} \tag{29}$$

and we wish to minimize the function:

$$[\frac{1}{n}\sum_{i=1}^{n} max\{0, 1 - y_i(w_i \cdot x_i - b)\}] + \lambda||w||^2, \tag{30}$$

where λ is a parameter that weights the trade-off of increasing the margin side so that the independent data vector x_i is on the correct side of the margin. Although the problem described here uses only two classes to try to define these margins, we can generalize these arguments to an n-dimensional hyperplane.

4 Data Sources

4.1 Classified Tweets

In order to train this model we had collect reliably classified tweets. In order to do this, we gained access to a data set of thousands of classified tweets from Axel Schulz and Christian Guckelsberger, which had classified tweets into multiple "categories": "fire", "shooting", "crash", and "NO" for neutral. We used a subset of these tweets that were associated with locations in Chicago, New York, Boston, Memphis, and Seattle, as the types of incidents could be assumed to be similar across these 4 metropolitan areas and the local vernacular for describing these incidents should vary a great deal across these American cities. Due to an imbalance in the number of incidents in the "crash" and "flood" categories, we decided to focus on classifying the "NO", "fire", and "flood" categories. [3]

4.2 Twitter API

In order to even classify tweets in the first place, we needed to pull tweets from Twitter. Since we are associated with an educational institution, we were able to pull 1500 tweets at a time every 15 minutes. Specifically, we tried to find tweets that were geo-tagged to help the ARC identify the location of incidents. These tweets would be stored on a MongoDB database, and could be pulled by query.

4.3 UCI Machine Learning Repository: Humanitarian Data

This is a sample of social media posts from Twitter and Instagram collected by researchers from the University of Beirut. We used some 600 tweets that classified fires and floods from these social media platform which should improve the reliability of the text classification model's performance. [2]

5 Results: Text Classification Performance

5.1 Scraper Alone

In early April 2019, Henry Post used our Twitter scraping tool to pull over 1,100 tweets geo-tagged 20 miles around Chicago without re-tweets in the data set that keyword searched "fire", "house fire", "mudslide", "landslide", and "emergency". We eliminated "landslide" and "mudslide" from the data set as Chicago is at an extremely low risk of having these events occur so close to Chicago. After eliminating this data, we manually parsed through every tweet to determine if it was a viable incident that the ARC could respond to. After over an hour of work, only a handful of tweets were classified as one of the categories used in the models. Although this did not take a significant amount of time or human-power to finish the task, if you remove the geo-tagged restriction on the scraped tweets this task becomes unfeasible.

5.2 Comparing Model Performance

We then evaluated the data we received from Schulz and Guckelsberger [3] along with the UC-Irvine data set [2] to train a model using each of the three methods described earlier in the paper using an 80/20 split of training

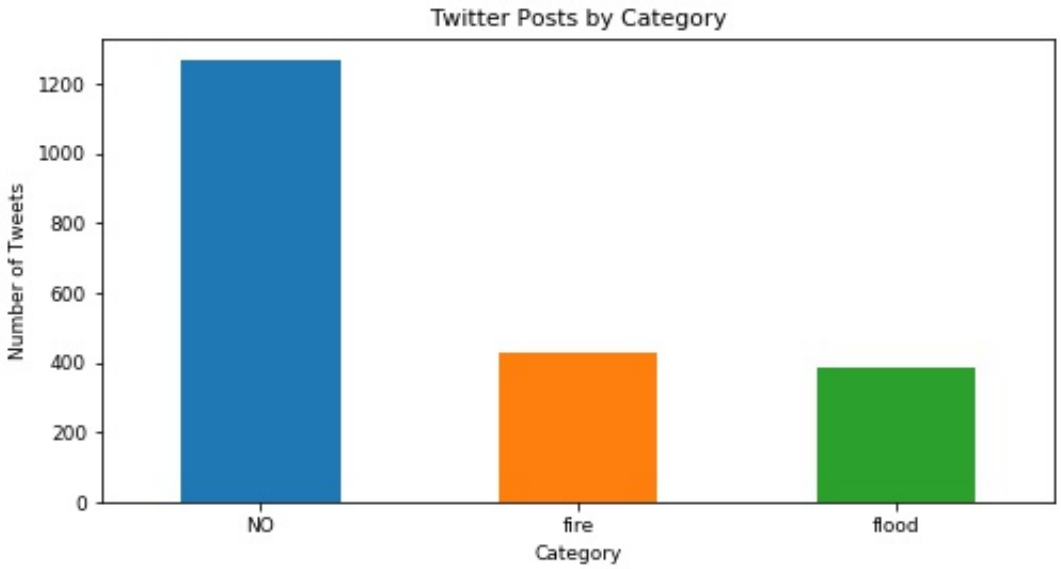

Figure 2: Number of tweets in each category used in training the model.

data and test data from these data sets. We additionally used a random forest classifier to try to classify the tweets.

There were four different types of models that were under consideration. Clearly, the random forest classifier was the least accurate classifier of the four models tested, probably due to the low amount of leafs used for the classifier. The LinearSVC model showed the highest accuracy, and thus we will choose to use the LinearSVC model to classify the tweets into the incident categories. (3)

The confusion matrix (4) shows not only the accuracy of the model, but also where the models misclassified the most, and what categories the models misclassified to. These results are based off tweets the model has never seen before. The model reads the text and outputs a prediction for that text, the bottom side labeled Predicted; either "fire", "flood", or "NO". Then it compares it to its actual label, the left side that says Actual. The matrix shows that out of all the tweets in the test data set, the model correctly classified 108 as "fire", 115 as "flood", and 412 as "NO", our neutral class.

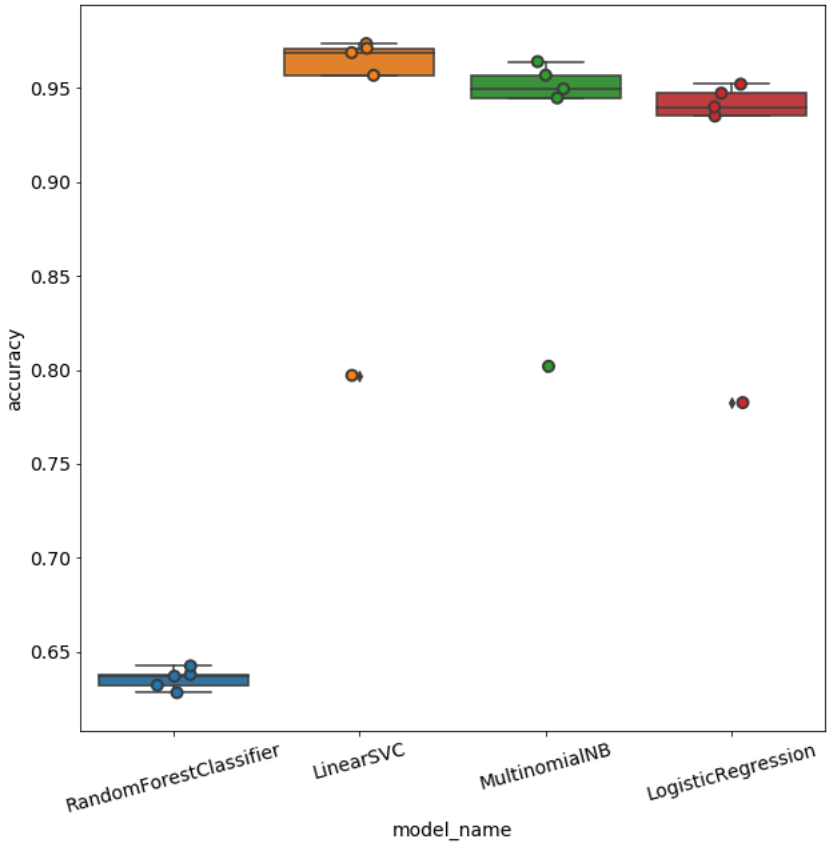

Figure 3: Results depicting the accuracy of the 4 classification models tested.

We chose to analyze the precision and recall scores of the models as these essentially measure the true positive rate of the model (6). Precision is equal to the true positive results over the divided by the number of true positive and false positive results, in other words the total number of correctly classified tweets over total number of predicted positive tweets. Precision gives an indication of the what percentage of tweets are returned by the model that are actually relevant to the search. Recall is the true positive over the total number of true positive and false negative results, or the total

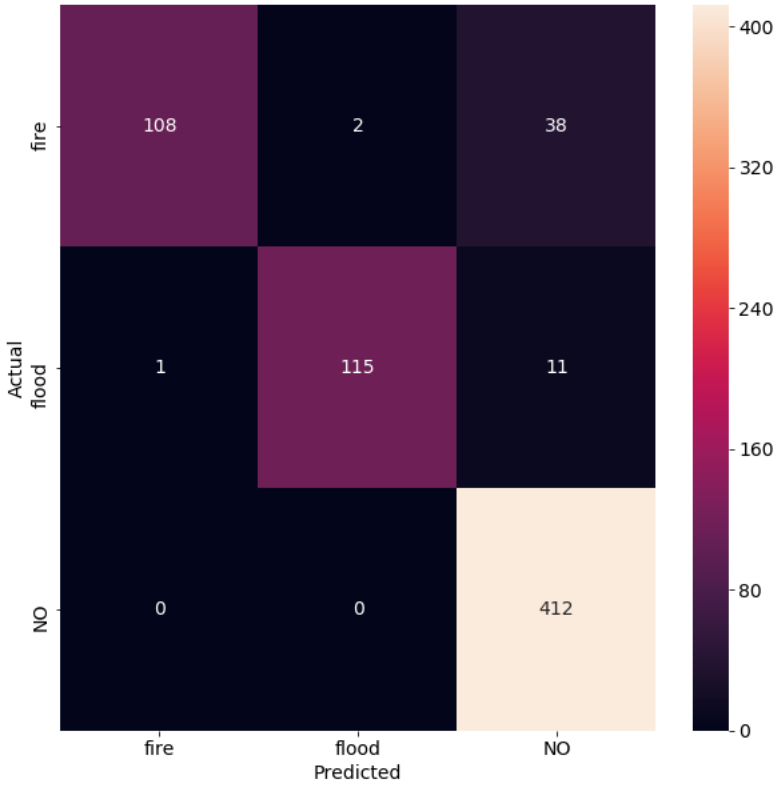

Figure 4: The confusion matrix for the data set being run through the model. The squares in the upper right of the diagonal represent the number of false negatives for each classification, the squares bottom left of the diagonal represent the false positives for each of the classifications, and the squares on the diagonal represent the number of true positives/negatives for each classification.

number of tweets that were actual positives over the number of correctly classified tweets. Unlike the precision, recall refers to the percentage of total relevant results correctly classified by our algorithm.

Actual ↓/ Predicted →	Fire	Flood	NO/Neutral
Fire	108	2	38
Flood	1	115	11
NO/Neutral	0	0	412

Figure 5: We see that the the LinearSVC model does well correctly classifying fire and flood social media posts. However, there is some difficulty in the classification of the neutral class.

	precision	recall	f1-score	support
fire	0.99	0.73	0.84	148
flood	0.98	0.91	0.94	127
NO	0.89	1.00	0.94	412
avg / total	0.93	0.92	0.92	687

Figure 6: Quantitative measurements of the model evaluating accuracy of categorical classifications.

The F1-score is defined to be:

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{31}$$

simply the harmonic mean of precision and recall. It could be seen as a weighted accuracy score, as it weights the accuracy if the categories are not balanced, which in this case they are. Overall the model performs decently given the current scores and the f1-score. A way to maximize the recall would be to perform the active learning part of the project, meaning it will begin to correctly classify tweets that are more similar to the ones the ARC would see, than just generic ones from our data set.

6 Conclusion

Over the course of the semester, this IPRO group was able to build a tool that could search Twitter by keyword or account to find potentially relevant tweets that the ARC could use for their incident response. Despite having the power of this tool, it was found to have too many irrelevant tweets to

manually parse through to be useful on its own. The research groups' work of finding machine learning classification algorithms to implement into the scraper proved successful in accurately and precisely analyzing social media posts from Twitter and placing them in the correct category. The analysis of the models that were tested led the group to select the 'LinearSVC' classifier to analyze the tweets. This classification model did not have any special features, thus it would take further research and fine-tuning to improve the current classification tool. The classification model needs to be integrated along with the scraper as well, in order to ensure the most efficient scraping and classification of tweets. Despite the amount of additional work needed to be done to have a fully functioning classification tool, in a barely a two months of work this group laid the groundwork for the ARC to implement a tool that it could use to increase the efficiency in its efforts to provide humanitarian aid to those in the Chicago area who need it the most.

7 Additional Future Work

7.1 False Positives and Active Learning Strategy

The general problem of active learning can be described in this case as follows: there is a set of tweets that our scraper will identify at some time i, where $i \in \{1, 2, ..., n\}$ and is a period of time, hour, day, week, etc. We have a set of total tweets, T. At each time period, we will have 3 types of tweets:

1. A set of tweets that already has a known classification, say this subset is called $T_{L,i}$, where the subscripts stand for labeled at time i.
2. A set of tweets that has an unknown classification, call this subset $T_{U,i}$.
3. A set of the tweets that has been picked to be labeled by the ARC or our program, say this subset is $T_{P,i}$.

The goal is to pick the best tweets for the third set $T_{P,i}$. In this case, the text classification model will have provided a first type of label, the type of incident the text classifier has associated with the tweet. Now, the ARC may be able to provide two types of labels to these tweets.

1. If the tweet's classification label was correct.
2. Whether or not the tweet was used in a response to an incident.

In response to the first item, our historical database of tweets is fairly large, so it seems unlikely that the classifications would be incorrect. However, given the prevalence of bots and other trolls that could produce seemingly authentic text, the ARC may want to identify these false positives. Second item is also of interest, as it would be ideal if the ARC would also provide the model with tweets that it found to be useful.

Although the issue with false positives is important, we can re-frame the issue to combine two problems into one. The model will be identifying tweets that are thought to be positive. We will make two assumptions. One is that the false positives and true positives pulled by our Twitter scraper/classification tool do not have the wrong incident type label. An incident could be a false positive, this is true, but we are assuming that if something has an initial label, say "fire", then it will not be reclassified as a different incident type, such as "flood". The second is that the that it is likely that the ARC would be able to find multiple reports of incidents when deciding on whether or not they should respond to the incident, from sources such as OEMC, IPN, or radioman911. The ARC contacts the local jurisdiction to see if they can be of assistance to an incident as well. This active classification would require an ARC volunteer to label the outputs from the scraper, and then to label the tweet as "True" or "False", i.e. the tweet was useful or the tweet was a false flag. This would be a scenario of stream-based selective sampling, as the tweets the model draws from Twitter are drawn one at a time from Twitter by the model [4]. The ability to confirm or refute an incident report is vital to being able to actively train the model.

From the tweets the scraper does receive, there will be a selection of which tweets the ARC uses. We propose the addition of these tweets to the database. The issue becomes if the ARC does not use tweets, but they are still true positives when identifying incidents, we do not want to simply label these as false positives. These classifications will "confuse" the model.

The decision whether or not to query new tweets that are pulled by the classifier can be handled in a few ways. One could try to query using an "informativeness measure, and make a biased random decision, such that more informative instances are more likely to be queried" [4].

7.2 Evaluating Reliability

Once the tweets are stored to our database, it may not only be useful to train our program to identify incidents from Twitter using new tweets/so-

cial media posts that it has labeled as corresponding to an incident in the Northern Illinois region, but it may be smart to keep track of accounts that are viewed as more useful in aiding the ARC's responses. This would perhaps require a ranking of accounts where accounts have been selected by the scraper, and a way to score the usage of these accounts' tweets for when they are selected for use by the ARC. For this, we propose two simple measures for these two account traits:

1. Actively keeping track of the number of times a certain account is identified by the Twitter scraper

2. Keeping track of a percentage of the number of times an account had their tweets used by the ARC when selected by our model out of the total number of times the scraper selects a tweet from Twitter.

This is not the most complex measure, but future groups could look at finding or developing measures of accuracy and/or precision to implement into the active learning analysis.

References

[1] Nachtheim C. Neter J. Li W. Kutner, M. *Applied Linear Statistical Models*. McGraw-Hill Education, 2005.

[2] H. Mouzannar, Y. Rizk, and M. Awad. Multimodal damage identification for humanitarian computing data set, 2016.

[3] A. Schulz, C. Guckelsberger, and F. Janssen. Semantic abstraction for generalization of tweet classification, 2015.

[4] B. Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009.