CS224N Final Project: Detecting Key Needs in Crisis

Tulsee Doshi (tdoshi), Emma Marriott (emarriott), Jay Patel (jayhp9)

March 22, 2017

Abstract

When a crisis occurs, the world springs into action to try and understand what is happening and what help is required. During these times, twitter has become a key avenue through which to disseminate information. In this paper, we classify tweets into 18 key categories (such as [volunteering] or [injury]) in order to organize the stream of knowledge coming in. We also attempt to predict *when* a particular category will be tweeted about by leveraging past tweets and categories to predict future tweet categories. Our most successful result, intermixing a Word2Vec model with a standard feed-forward classifier is able to classify the tweets with 59% accuracy. We find that more sophisticated models (including added depth to a feed-forward model and an LSTM that uses a sequence of words to predict a tweet) prove to be less effective than a more generic classification methodology, likely because of overfitting on labels that are more common and 2) noise created by combining various types of crises, such as earthquakes and floods, together. We show that our classification model is more successful when restrained to specific types of events, with 70% accuracy on earthquake data, for example. Additionally, we determine that leveraging past history to predict future tweet patterns is moderately successful. We showcase our LSTM methodology which has achieved a 33% accuracy. Lastly, we test out our model on the more recent twitter compilations for two natural disasters: the earthquake in Italy and Louisiana Floods. We show, via qualitative examples and analysis, that our model can likely scale to natural disasters of various forms.

1 Introduction

2016 saw a number of high-profile natural disasters ranging from the devastation in Haiti, to the earthquakes in Italy and Japan, to the floods in China. According to CNN, 2016 has seen the largest financial loss as a result of natural disasters – a whopping $170 Billion dollars [5]. In the US alone, natural disasters have taken 8 million lives since the year 1900 [3].

Management of natural crises is non-trivial. In most cases, connectivity drops, infrastructure breaks down, and phone line jams block engagement. As a result, the world is often standing by at a distance, unaware of the exact state of the situation or how individuals can jump in to help. In the modern environment, **Twitter** has taken on a critical role in crisis management. Local tweets provide updates about injuries etc. from the actual location. Tweets from volunteers, non-profits, and other organizations provide details about community involvement and donation ability. While these tweets are grouped together by a collective event hashtag, they often emerge as a disorganized stream from which important details can be easily lost.

Tangentially, mining twitter data for information has become a key application for natural language processing. New approaches to understanding words in an embedding space (such as word2vec and glove), have led to an increased understanding and an ability to classify tweets based on a small vocabulary. Additionally, both traditional techniques such as SVMs as well as more advanced neural networks and sequence models have been applied successfully to tweet-classification problems. These successes are discussed in more depth in the background section below.

In this paper, we tackle crisis management and the organization of information, via classifying relevant tweets into key categories. Some examples are listed below:

- Donation offers or volunteer services
- Displaced people and activations

- Animal management
- Response efforts
- Caution and advice
- Sympathy and emotional support
- Injured or dead people
- Missing, trapped, or found people
- Infrastructure of utility damage
- Diseases
- Not related or irrelevant

We begin by applying generic linear regression models to the classification problem, and improve upon this by applying more sophisticated techniques, implemented in Tensorflow. Further details are provided in the 'Approach' section below.

2 Background / Related Work

Classification of tweets has been tackled in many shapes and forms over the years. Overtime, we've seen a shift from one-hot vectors representing words to more dense vectors based on word embeddings. Yang et.al, for example, show how leveraging word embeddings can improve classification of tweets to predict election results [7].

Crisis response, in particular, has been tackled leveraging twitter data as well. The paper from which we borrow data, by Imran et al, focuses on building a strong word2vec model based on crisis response tweets and leverages basic linear regression models[2].

More sophisticated models have not been used heavily in this space, though a couple papers have been published in the last year which leverage LSTMs to improve text classification accuracy. Most notably, Zhou et al showcase that c-LSTMs, a hybrid approach between CNNs and LSTMs showed significantly improved results over traditional models when classifying text [8].

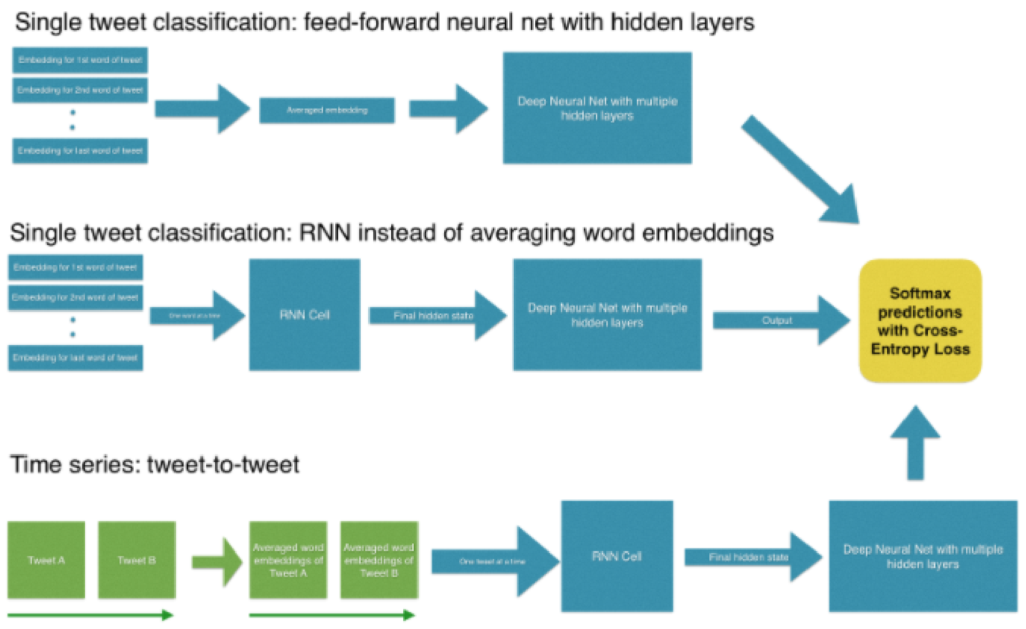
3 Data

In this paper, we use data created by Imran et al for their paper, "Twitter as a Lifeline: Human-annotated Twitter Corpora for NLP of Crisis-related Messages" [2]. The dataset consists of tweet IDs from 19 different natural disasters (including the Nepal earthquake and Ebola virus outbreak). These 19 disasters represent 8 types of crisis: Earthquakes (5 datasets), Typhoons (3 datasets), Volcanoes (1 dataset), Floods (2 datasets), War & Conflicts (2 datasets), Biological (2 datasets), Landslide (3 datasets), Airline Accident (1 dataset). Not all of these datasets are in English, however. We test with both combining Spanish & English and only leveraging English data. We determine that only using the English datasets is most effective. The data we use also includes volunteer-labeled as well as crowd-labeled classifications such as "injury_and_death" or "volunteer". There are 13 such categories.

After taking into account tweets that are no longer present on Twitter, gaps between the labeled data and the set of IDs, as well as different languages, we conclude with a resulting data set of 39000 labeled tweets. We split these tweets into an 80% training set and 20% test set.

3.1 Pre-processing

Using the tweet IDs and a twitter API (tweepy)[6], we scrape tweets based on provided IDs directly from Twitter, along with the time at which they are posted. We then remove punctuation and stop words and replace common twitter slang with complete terminology so as to create consistency (for example, modifying 'gr8' to be 'great').



Single tweet classification: feed-forward neural net with hidden layers
Embedding for 1st word of tweet
Embedding for 2nd word of tweet
Embedding for last word of tweet
Averaged embedding
Deep Neural Net with multiple hidden layers
Single tweet classification: RNN instead of averaging word embeddings
Embedding for 1st word of tweet
Embedding for 2nd word of tweet
Embedding for last word of tweet
One word at a time
RNN Cell
Final hidden state
Deep Neural Net with multiple hidden layers
Output
Softmax predictions with Cross-Entropy Loss
Time series: tweet-to-tweet
Tweet A
Tweet B
Averaged word embeddings of Tweet A
Averaged word embeddings of Tweet B
One tweet at a time
RNN Cell
Final hidden state
Deep Neural Net with multiple hidden layers

Figure 1: Diagram illustrating the 3 models: feed forward, LSTM based on individual words, and LSTM based on tweets

3.2 Word Embeddings

Previous research has shown that dense, learned, word embeddings perform better than sparse one-hot vectors. We therefore optimize high-quality embeddings. We begin with a baseline of two sets of Glove pre-trained vectors from the Stanford NLP toolkit[4]. The first is a set of 6B tokens trained from Wikipedia, and the second is a set of 27B tokens trained specifically on twitter data.

We then improve upon our word vectors by training word2vec skipgram models upon on our training data of tweets. As shown in Yang. et al, we expect that training specifically on these words will allow for a more sensitive vocabulary, and will more accurately capture the embedding space for the crisis response vertical.

In our simple feed forward models (described below), we model each tweet as a simple average of its word embeddings. While some papers, such as De Boom et al[1], note other ways of doing this such as concatenating the min and max of the individual word features, we find that averaging the words proves to be the most successful.

4 Approach

4.1 Baseline based on previous paper

We begin by replicating the work of Imran et al, from whom we take the data. We train an SVM, a Random Forest, and a Naive Bayes classifier and use this to predict tweet categories.

We penalize the SVM with the L1-norm in order to account for the high likelihood of having non-linearly separable data, given that the data is potentially erroneous. Additionally, we start with a baseline Radial Basis Function (RBF) kernel. The RBF kernel references the following equation: K(x,x') = exp(||x-x'||2). With our RBF kernel, we use the recommended baseline C value of 1.0 and Gamma value of 0.025 (1/ the number of features).

4.2 Simple feed forward classification

As an initial step to improve classification, we create a simple feed forward neural network. The network functions as shown in the first row of Figure 1. The individual word embeddings are averaged and inputted in batches to a neural network and implemented in Tensorflow. We test a variety of functions, but find best performance with RelU nonlinearities and gradient descent. The final prediction is based on a softmax distribution.

We experiment & tune for optimal performance. We first look at various levels of hidden layers (ranging from 1 to 10), and determine that 1 level of depth, with dimension of 200, is the most successful. This is likely because more layers on such a small dataset, lead to over-fitting.

Figure 2 below shows the tuning of the learning rate. We see that a learning of $3.0 * 10^-5$ is the most optimal for reducing loss and improving overall performance on the simple feed-forward neural net, our best performing model overall.

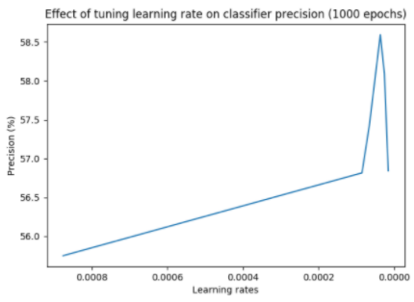


Figure 2: Learning Rate vs. Precision

4.3 LSTM classification

We build upon the feed forward baseline by adding an LSTM classifier at the base (shown on the second row of Figure 1). This classifier takes sequences of word embeddings for a sentence and produces a hidden layer output at the end of each sentence. Although each tweet is truncated/padded to be 20 words in length (various padding amounts were tested), we use tensorflow's dynamic_rnn function which dynamically generates the computation graph given variable sentence lengths (up to our maximum length). We also experimented with stacked LSTM cells, although we found that this multi-cell configuration did not further improve accuracy. The hidden layer from our LSTM is then fed into a DNN with multiple ReLu layers. We found that the best performance was given by just one layer with input size 64. The layer is then passed into a softmax distribution with cross-entropy loss. The equations for the LSTM and its structure are given in Figure 3.

$$
\begin{aligned}
i_t &= \sigma(W^{(i)}x_t + U^{(i)}h_{t-1}) \\
f_t &= \sigma(W^{(f)}x_t + U^{(f)}h_{t-1}) \\
o_t &= \sigma(W^{(o)}x_t + U^{(o)}h_{t-1}) \\
\tilde{c}_t &= \tanh(W^{(c)}x_t + U^{(c)}h_{t-1}) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}
$$

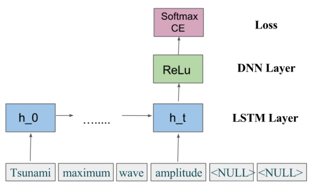


Figure 3: LSTM Hidden Layer Update Equations (left) and Model Diagram (right)

Once again, we experiment and tune for various parameters. Figure 4 shows the loss curve from a model with one LSTM and two hidden layers. The table below shows training accuracies and test accuracies with and without attention on the earthquake data set. Although attention performs better in training, it clearly over-fits and actually performs worse than no attention in test accuracy.

4.4 Classifying tweets in a time-sequence

Finally, we attempt to classify the label of a tweet by building a sequence model based on the previous tweets (based on time posted). This is the 3rd row of Figure 1. Many crises have consistent arcs: damage –> information about injury –> sympathy and solidarity –> relief efforts –>

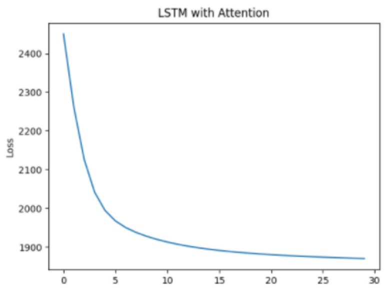


Figure 4: LSTM loss

Attention	Training Accuracy	Test Accuracy
None	51.65%	45.62%
Length 3	57.48%	42.18%

Table 1: Results from model outputs

volunteering and donations. We hope that this consistency will allow us to build stronger context about the tweet in question and therefore learn more thoroughly.

This sequence model thus differs from the previous in that instead of taking in individual words in a single tweet, the model inputs a sequence of tweets one a time, and uses these as well as the tweet itself to predict the label. The word embeddings in this case are either averaged or aggregated using min-max (a concatenation of the minimum and maximum embeddings across each dimension), such that embeddings can still be learned on during training.

As in the other models, we experiment with different levels of attention, hidden layers, learning rate, and optimization method. We find as before that hidden layers do not further improve accuracies, while adding attention here helps slightly. Our best performing model had an attention length of 3, a tweet sequence length of 6, and two ReLu layers in its DNN component. The loss curve and confusion matrix are shown in the figure below.

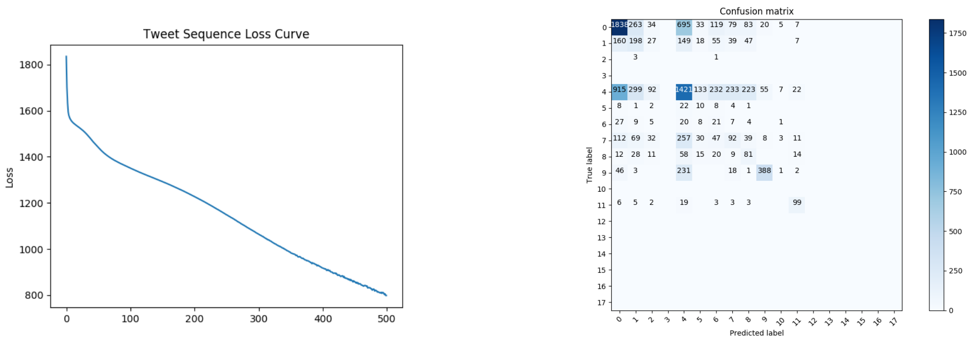


Figure 5: Loss curve (right) and confusion matrix (right) for tweet sequence training

4.5 New Twitter Data

In order to test the scalability of our model, we scrape 200 tweets about the Louisiana Floods and Italy Earthquake last August. We run our model on this sample, and qualitatively evaluate a sample of 25 tweets. These results can be seen in the Experiment section below.

5 Key Experiment Results

Our initial baseline methods prove to severely under-fit the model. Table 1 showcases the precisions for each of the SVM, Random Forest, and Naive Bayes models. When tested on the training

Model	SVM	Random Forest	Naive Bayes
Precision	31.25%	31.25%	25.00%

Table 2: Results from model outputs

data, we see that the output is similar, indicating that the models are training quite poorly.

We then train and run each of our models on all of the English data and compare their best output. This is shown in Table 2.

Additionally, we compare the 3 types of word embeddings to evaluate their relative performances. We find that the pre-trained glove vectors have a disadvantage in that many of the words in our tweets are not in the vocabulary of these vectors. Thus, the trained word2vec embeddings perform the best. This can be seen in Figure 6.

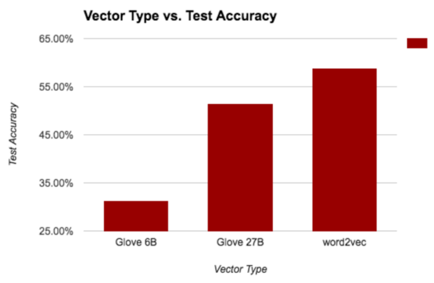


Figure 6: Word Vector Types vs. Test Accuracy

Model	Feed Forward	Feed Forward	LSTM (word sequence)	LSTM (tweet sequence)
Precision	58.50%	59.01%	45.06%	43.9%
F1-Score	.26	.33	.34	0.31

Table 3: Results from deep-learning model outputs

We see that the feed forward with a single hidden layer performs the best, while the two LSTMs perform more poorly than expected. Additionally, the tweet-to-tweet sequence, despite having the most context, performs the worst in classifying our data. Our hypothesis for this is that more parameters require more data to learn properly, so any more complex model suffers. With a bigger data set, we believe an LSTM can outperform the feed-forward neural net.

We investigate this more fully by looking at a confusion matrix to better understand what is being predicted correctly or incorrectly. In the confusion matrix, which is for Earthquake data, we can see that not all labels are in all event types. The Earthquake data, for example, only has 6 label types. Additionally, we see that some labels occur significantly more frequently than others. 'Other_Useful_Information' and 'Not_Relevant_Information' are the only two labels, for example, that span every event type. They are also the two most common categories, with 51.56% of the labels being 'Other_Useful_Information' and 7.10% being 'Not_Relevant_Information'. The large percentage emphasis on one label type likely leads to over-fitting on that label. **Note:** in the confusion matrix, label 0 corresponds to 'Not_Relevant_Information' and label 4 corresponds to 'Other_Useful_Information'. The most common misclassification was 'Not_Relevant_Information' as 'Other_Useful_Information - in examples such as "Balochistan tragic earthquake killing more thn 300 people PTI Balochistan," the classifier seems to have given the correct label (given that this pertains to the Pakistan Earthquake), in spite of it being labeled as 'not relevant information.'

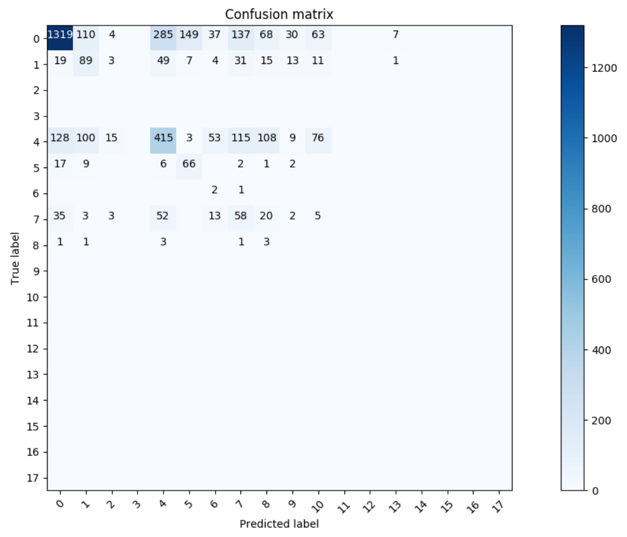


Figure 7: Confusion Matrix for Earthquake Classifications using tweet sequence LSTMs

Model	Feed Forward	LSTM (word sequence)	LSTM (tweet sequence)
Precision	66.30%	71.01%	51.75%

Table 4: Model outputs on only earthquake data

Looking through more examples, we found that the disparities in the frequencies of these labels, as well as unreliability from some volunteer labels, complicate the learning task.

Interestingly, we see that when training and running the model on a single event type (ie. only Earthquakes), the model performs much better. We believe this is the case because various event types have specific labels associated with them that may not be present in other events. Additionally, various events may have different time-series arcs, as well as different word usages. These differences may make it more difficult to detect patterns and also make it easier to over-fit on the popular labels that span across multiple events. This was contrary to our hypothesis all along that a model should generalize to different natural disasters with ease, which rested on the flawed notion that tweet content related to 'money' or 'asking for help' (labels that showed up across a range of disasters) should not differ all that much across disasters.

Finally, we run our feed forward model on the Louisiana Flood and Italy Earthquake data. We qualitatively analyze the outputs, and find that in the sample of 25 tweets, 12 are classified correctly, implying a precision of **0.48%**. Some examples of classified tweets are below:

- *"Tethered and Abandoned Dogs Left to Die in Louisiana Floods Are Now Forever Safe"* – [Animal Management]
- *"Severe Weather Awareness Week in Louisiana is a good time to prepare for floods, here's some tips when prepping* – [Caution and Advice]
- *Earthquake (#terremoto) M2.1 strikes 29 km W of Ascoli Piceno (Italy) 10 min ago.* – [Other Useful Information]

6 Conclusion

In conclusion, we find that our neural net model outperforms the linear classification approaches previously used in this domain. We also find that our model scales moderately well to events that were not in the training set (such as the earthquake in Italy). We do find, however, that non-linear, more complex approaches struggle to perform as well. We believe this to be the case because of a few combining factors: 1) The small sample of usable data, 2) Overfitting to the most common labels like "Other Useful Information" or "Not Relevant Information", and 3) Noise from combining multiple different types of events together.

This study accentuates the importance of having larger, more complete datasets, as well as the importance of training on events that are classified similarly to minimize confusion in the model. Additionally, we learnt that over-fitting can happen at micro-scale, on a particular label or set of labels.

A future effort could attempt to retrieve crisis related data from not just Twitter, but also other social networks like Facebook and Instagram. Additionally, we see better performance when running the model on individual crises than when training on an aggregate dataset, which breaks our initial hypothesis that tweets across crises should be quite similar given the nature of natural disasters. To further explore the correlation inherent in the data (or lack thereof), it would be interesting to train a model to predict what crisis a tweet is talking about. If that performs well, it implies that the tweets for different crises are substantially different so we should not expect our very general neural net (of 59% precision) to do much better.

7 References

[1] Boom, Cedric De, Steven Van Canneyt, Thomas Demeester, and Bart Dhoedt. "Representation Learning for Very Short Texts Using Weighted Word Embedding Aggregation." Elsevier (2016): n. pag. Ghent University, 2 July 2016. Web. 21 Mar. 2017.

[2] Imran, Muhammad, Prasenjit Mitra, and Carlos Castillo. "Twitter as a lifeline: Human-annotated twitter corpora for NLP of crisis-related messages." arXiv preprint arXiv:1605.05894 (2016).

[3] "Natural Disasters since 1900-over 8 Million Deaths and 7 Trillion US Dollars Damage." Phys.org - News and Articles on Science and Technology. Phys.org, 18 Apr. 2016. Web. 22 Mar. 2017.

[4] Pennington, Jeffrey. "GloVe: Global Vectors for Word Representation." GloVe: Global Vectors for Word Representation. Stanford University, Aug. 2014. Web. 21 Mar. 2017.

[5] These Disasters Helped Push the Total Damage Caused Natural Catastrophes to 175*Billionin*2016.”*NaturalDisaste*
Billion in Damage in 2016." CNNMoney. Cable News Network, n.d. Web. 21 Mar. 2017.

[6] "Tweepy." Tweepy. N.p., n.d. Web. 22 Mar. 2017.

[7] Yang, Xiao, Craig Macdonald, and Iadh Ounis. "Using word embeddings in twitter election classification." arXiv preprint arXiv:1606.07006 (2016).

[8] Zhou, Chunting, et al. "A C-LSTM neural network for text classification." arXiv preprint arXiv:1511.08630 (2015).

8 Teammate Contributions

Jay, Emma, and Tulsee worked collaboratively to put together this project. Tulsee created the initial baseline models and word2vec model, Jay created the initial feed-forward model, while Emma created the LSTM. Each of these was then iterated upon and run by the other two team-members in terms of tuning, adding hidden layers, attention, etc. The paper was written by Tulsee, with editing and additions by Jay and Emma.