Tweet Classification Study

COMP 380 Neural Networks

James Pala
University of San Diego
apala@sandiego.edu

Marcus Rogers
University of San Diego
marcusrogers@sandiego.edu

Taylor Wong
University of San Diego
taylorwong@sandiego.edu

ABSTRACT

In this project, we explored a data mining and machine learning study tweet topic classification. This is a sparse-input classification problem that we will attempt to use a neural network to solve. Specifically, we explored Tweet topic classification using different crisis events collected between 2012 to 2013. After removing stopwords, using a shallow 3-layer back-propagating network architecture with a large input vector size, we were able to achieve ~99% training and testing accuracy.

Keywords

Natural language processing (NLP), Twitter trends, text classification, text processing, neural network.

MAIN REPORT

1. Introduction

As we attempt to design more general artificially intelligent systems, comprehending and responding to natural speech is a requirement. A goal for computer systems in the short term is classification of documents, which allows a system to understand the topic of a document. Natural Language Processing (NLP), which text classification is a subfield of, is an inherently difficult field, as computer systems are not well suited to deal with human-generated text data. As a result, preprocessing techniques of the data must be used.

Specifically, our model attempts to classify Tweets, which presents a unique problem due to their small size. This means that the input vectors to any model will be sparse. Additionally, Tweets are noisy and may have information unrelated to the categories you are classifying, as well information that does not contribute to the classification of the Tweet. However, if Tweets can be accurately classified, then Twitter trends can start to be analysed. This intelligence has an impact in fields from business, in company and sector analysis, to politics, in election prediction. Even further, new events could potentially be identified, which could help authorities respond quickly, or help news organization learn of new developments quickly.

We applied a shallow neural network on tweets about various crises. Using several data preprocessing techniques before feeding the inputs into a backpropagating net implemented with Keras API, we were able to achieve >99% training and testing accuracy. We tested the model on several layers of abstraction of our data, as well as validated with tweets about previously unseen events. For more information, see the section 3.4.

2. Background

Natural language processing (NLP) has many subfields and approaches. Showing computers how to deal with language input, or even understand language, is extremely relevant to allow automated systems to interact naturally in a dynamic, human environment. Our research revealed many general statistical as well as machine learn techniques and experiments that attempted and, in many cases successfully, achieved NLP tasks. Some examples include the IBM Watson NLP API. It can analyze the semantic features of text input, including categories, concepts, emotion, entities, keywords, metadata, relations, semantic roles, and sentiment. The link for a demo of the API can be found in Appendix D. Other examples include the Reuters Tracer program, which uses (non-neural network) clustering techniques to identify and rate the newsworthiness of events as they occur using Twitter data. It claims that it was able to identify the San Bernardino shooting 7 minutes before traditional news sourcing techniques, and often is able to recognize newsworthy events up to 60 minutes before other sources.[8] Other research includes a study which used Twitter analytics and statistical machine learning techniques to identify riots before they occur. The paper claims it was able to predict the Arab Spring riots in Egypt.[7]

While the above examples, as well as other research we did, showed significant research has been done on NLP tasks using machine learning models such as support vector machines. However, neural network research on the subject was significantly more limited, particularly regarding Tweet classification. Because of this, we focused on a simple neural network approach that would serve as a proof of concept to Tweet classification. Fortunately, the text and data preprocessing techniques are the same, regardless of model type. We researched many different preprocessing techniques and implemented the ones we thought would be most effective in our attempt to classify tweets.

3. Approach

We used a prelabeled set of over 25000 Tweets on natural disasters. We converted these tweets as well as their labels to formats that could be processed by the net, then fed them to a shallow, backpropagating net with one hidden layer to attempt to classify the tweets.

3.1 Data Source and Representation

For this application, we used supervised data and labeled it by specific categories of crises. We had multiple data sets

categorizing the tweet either by location and event event. The data set categorized by location and crisis included 25 different labels (See Appendix D). Additionally, in the dataset with 25 different crisis, there is about 900 tweets per category. The data set categorized by only crises had 15 different events. For each of the distinct data sets, we created a script to relabel the original data we collected and save the newly categorized data into a separate file. Another way of diversifying the data and challenging our neural net was to test the net with data it has not seen before. For testing, we created a separate data set with similar crisis events, excluding duplicate tweets.

We generated a vector of max features which revealed a unique challenge: Extremely sparse input vectors because there are a limited number of words in a tweet but a high number that occurs in a set of tweets (corpus).

3.2 Preprocessing

In order to prepare the data for classification, it had to be preprocessed. The four main methods of preprocessing we researched and utilized were TF-IDF vectors, count vectorizers, stemming, stop word removal and N-grams. TF-IDF is the most popular preprocessing technique and is short for term frequency-inverse document frequency. It is a numerical value that determines how valuable or important a word is in a tweet. A TF-IDF vectorizer generally punishes words that are common in a document or corpus. We did not use this approach to preprocessing our data because we feared that it could unintentionally punish words that we deem important in the corpus. For example, the word "fire" would occur often, since many of our texts referred to forest fires, yet this word is important to the correct classification of these tweets, especially due to the sparse input size. We wanted to avoid punishment of important words such as this. Instead we removed stop words to reduce the noise brought about by common words that did not contribute to the classification. Stop word removal is a basic preprocessing technique that aims to remove noise and unnecessary words from each tweet. Some example stop words are "the" and "a". Unlike TF-IDF vectors that would just lower the weight of these words, this method completely removes them while keeping the meaning of the tweet the same. A higher TF-IDF value indicates a more important word. The count-vectorizer simply tokenizes a tweet and then counts how many times a certain word appears in the tweet. We also researched stemming as well as n-grams, but did not implement these techniques because we were able to achieve good results without them. Stemming is used to reduce the number of different forms a word can exist as. For instance, the terms "car", "cars" and "car's" would become car. This helps simplify the text while keeping the original meaning of the text the same. N-grams are a combination of adjacent words with length n. Depending on the "n" value, they create sets of words from a tweet. N-grams are typically used to predict the next word of a sentence. They also help clump words together that are typically always adjacent to each other. For instance, the words "high school" and "San francisco" generally co exist with each other. Another preprocessing tool we did use is the label binarizer. This allows us to represent the different categories or outputs as binary vectors. For instance, the label 1 would have the corresponding vector [1,0,0,..,0] and label 2 [0,1,0,...,0]. In other words, as the data is fed through the net, it can only be categorized as one event. It is either the event or it is not.

3.3 Techniques

After removing stop-words from the data set, we split it up a 75% training, 25% testing split. We used the Keras API to implement a three layer, sequential, backpropagating net. This net used the Adam optimizer, with the Mean Squared Error loss function. Weights from inputs to any hidden layers (when tested) as well as to the output layer were set to nonnegative. Our input layer was of size 100000, and our output layer was the size of the number of possible categories we were classifying to.

The Adam optimizer was used because it is a recent, recommended extension to the classic stochastic gradient descent. the Mean Squared Error (MSE) loss function is a standard loss function which seeks to minimize the mean loss between the actual and predicted values, where:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(f_i - y_i)^2$$

where N is the number of data points,
f_i the value returned by the model and
y_i the actual value for data point i.

The weights of inputs to hidden layer and hidden layer to outputs were restricted to be nonnegative. This was because our net did not seek to find semantic meaning in tweets. Instead we sought to classify tweets based on topic or event type. In this case, we want to look out for certain words that would indicate the tweet was referring to a specific event or type of event. Because these significant words, in most cases, only indicate an event, and do not detract from classification, all the weights should be nonnegative. (If a text is talking about how the forest fires are gone, the topic is still forest fires, despite it being about being the absence of them). If our net had sought to understand or parse semantic meaning, negative weights would have been justified, for example with words such as 'not', which might have needed a negative weights because of its negative meaning. Finally our input size, or max features, was set rather large, at 10000. This means the 10000 most common words in the data set were inputs in the vector. After experimentation, we discovered that a higher input vector size allowed the net to be more accurate. This is because it has more data points to make a classification. We did, however, remove stopwords. While this was not entirely necessary (we achieved similar results while they were still in the data set), we did want the net to be more versatile and have the potential to handle higher amounts of data. Removing common words allows it do that because those stopwords do not help further classify a tweet, and allows the net to have more relevant information as inputs.

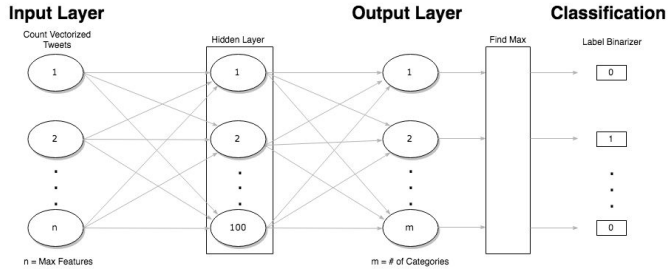
We usually set the number of training epochs to 3, as more epochs did not affect the accuracy significantly, and too many brought down the accuracy. Finally, after adjusting the batch size, we settled the batch size at 32. This is the number of samples before the net back-propagates the error. A larger number allows it to generalize more and reduces overfitting. Too high, however, results in the net generalizing too much and is unable to correctly classify much of the test set. Finally, the activation

function was a simple linear, although experimentation with other activation functions yielded similar results.

Finally, the output values of the net were float values, results of the linear activation functions. We needed the output in a vector form the size of the number of categories, with only one node at 1 and the rest set to 0 (since the tweet could only have one classification). The output node with the largest value corresponded to the classification, so we set up a find max function. Since this max node corresponded to the classification, we set it to 1, and all the others to 0. This was because our labels for the tweets were label binarized vectors.

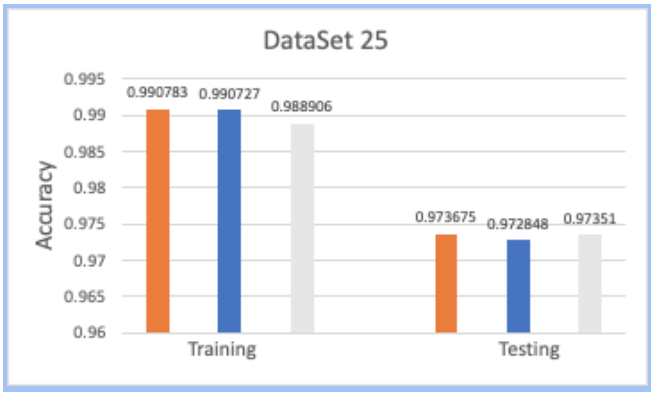
To summarize, the net took in a count vector of size 10000, fed it through a back-propagating neural net with the Adam optimizer on the MSE loss function with one hidden layer. The output node were float values, of which we found the highest node, which corresponded to the classification.

A general schematic of our network, see Appendix D for a larger version:



3.4 Results

When testing and training on 25 different crisis from different parts of the world, our training accuracy was ~99.07% and our testing accuracy was ~97.36%.



When we reduced the data to 15 categories based on the type of event (ie made the labels represent the abstract crisis, and include multiple instances of the crisis) the network achieved similar results. In fact, the test sets were even more effective, at ~99%.

In our final experimentation of this text classification net, we when we created a validation of events that it had never seen before, the net was only able to classify them with about 35% accuracy:

Validation print out: "Test: 2150 out of 6099 correct."

This validation set consisted of events that were the same type of events that the net had trained on, but were new instances of the crises in new locations that it had not seen.

3.5 Analysis

We optimized the hyperparameters of the network for this sparse input vector text classification problem. The explanation for why different hyperparameters where given different values is above in section 3.3. To review, however, the important features of our network were the nonnegative weights, because words cannot detract from a classification only contribute (especially as we are not seeking semantic understanding). Additionally, our input vector, expressed as max features of the countvectorizer was high, at 10000-size input vector of the 10000 most common words in the data set. This input size was relatively large because the higher this number, generally, the more accurate our testing and training became. For further explanation of the hyperparameters see section 3.3. However, an explanation of the results is needed:

After finding the best hyperparameters, when fed the data about 25 separate events, our net achieved a testing accuracy of ~97%. The accuracy improved the larger the input vector, so it can be assumed that a larger input vector size is needed for data sets with categories to classify. However, with this set, we were not able to increase the training accuracy above 97%. This is mostly likely because within the data25 set, there are many similar tweets, since many are referring to the same type of event, but in different locations, for example one label corresponds to tweets about a forest fire in Colorado and another corresponds to tweets about a bushfire in Australia. Additionally, some refer to different events in the same location, for example floods in Colorado and fires in Colorado. This ambiguity in the input data makes it hard for the net to classify with closer to 100% accuracy specific instances of events in different locations. Despite this, at ~99%, the results are pretty impressive.

Furthermore, when we trained on more general event data, the test accuracy of the network increased to the ~99%. When we trained on the data 15 set, which labeled similar events the same (eg floods were floods, regardless of where they occured), the accuracy of training and testing both were ~99%. This is most likely because the categories were more broad and contained more

data for each set. For example, the 3 examples of floods were now all in the same set. Not only did the net have 3x the data to train on for that category, but also had 2-3 different instances of the same type of event in different locations with different tweets about them. This allowed it to get a more general understanding of what referred each of these types of events.

While it was extremely good at tweet-level generalization for events it had seen before, when we fed the pretrained net tweets about events it had not seen, it was able to classify these with only a ~35% accuracy. This shows us that the net is not very good at event-level generalization and it cannot recognize new, similar events to ones it has trained on. This could be an advantage, if the goal of the net is to identify a new, unseen event. This could be the case in a net that had the goal of recognizing new crises or events as soon as possible for the authorities benefit, or to allow for speedy reporting (similar to the Reuters Tracer system). We think the reason for this lack of ability to generalize on the event-level is because the net overfits on the data about events that it is trained on. It is still able to recognize new inputs about the same events, but it overfits on these events. A solution to this would be training on data that has no location-specific data, or implementing a model architecture that prevents this overfitting, such as deep neural network with dropout.

4. Conclusion

In summary, this shallow neural network takes tweets as inputs and removes stopwords, then transforms them into a sparse vector using countvectorizer technique. Once fed through the network, the max output is found to find the classification the network assigns to the tweet. We achieved 99% accuracy with tweets about previously seen events, and ~35% accuracy with tweets about previously unseen events.

Several conclusions can be drawn from our results:

First, while our set of training/testing data was large, >24000 samples, it was still relatively small in comparison to the >500 million tweets generated per day and >500 billion per year. The authors had neither the storage nor processor power to approach this level of data, and so was beyond the scope of this project. However, this means our data and subsequent results represent a relatively small portion of the data and events available to classify. As such, this is more of a proof of concept of event classification of tweets. Good results came out of this, as the network was able to classify specific events with an extremely high degree of accuracy. Additionally, it was also able to generalize on an topic level about similar events, as long as it had seen the event before, to an even greater degree of accuracy. We included several techniques in our network that would allow it to handle higher levels of data, such as stop word removal, but further improvement is needed to increase the scale.

Our network was extremely good at memorization of events. While the tweets themselves in the testing set were new and unseen by the net, the events they described or referenced had been seen by the net. The good results were actually quite surprising. However, when we took it a step further and asked the net to generalize even further, it did not perform well. We trained on a variety of events, then fed it validation data of tweets about events it had never seen before. Although it had seen similar events in the training, the validation set had new instances of those same types of crises that it had not been trained on. When validating on these unseen event tweets, the net achieved only ~35% accuracy. We can draw the conclusion that network must see an event in the training data before it can classify similar, unseen tweets about the event. We have shown the network is good at tweet-level abstraction, but not so good at event-level abstraction.

4.1 Future Work

Due to the broad applications of text classification and as well as different variety of categories to classify, there is much future work that could be potentially explored. Specifically with tweet classification, the next steps include implementing a stemming program as well as a TF-IDF approach to the input data. We also hope to experiment with different network architectures and types of networks, for example convolutional neural networks and other deep network types. Our goal would be to increase the event-level generalization ability of the network.

Additionally, we would like to apply this or similar networks to sentiment analysis. The problem of text classification is similar, but is more abstract. Furthermore, since in reality twitter data is full of noisy tweets unrelated to the topics the be classified, we would like to implement a noise-reduction step before the tweets are fed to the network and/or the network would be able to recognize unrelated data. This perhaps could be achieved through a threshold, or other methods.

Finally, we would recommend setting up the network to a Twitter API pipeline to classify realtime data. Eventually, the goal would be to recognize trends. Additionally the goal would be to identify events as soon as possible as the occur to benefit the authorities, as well as attempt to predict possibility of occurrences of events before they occur.

5. Acknowledgements

Our thanks to Dr. Jiang for a wonderful semester and truly engaging class on one of the more interesting aspects of our field.

6. References

[1] Romero, Simone, and Karen Becker. “A Framework for Event Classification in Tweets Based on Hybrid Semantic Enrichment.” *ScienceDirect*, Academic Press, 15 Oct. 2018, www.sciencedirect.com/science/article/pii/S095741741830678X#sec0016.

[2] Bansal, Shivam. “A Comprehensive Guide to Understand and Implement Text Classification in Python.” *Analytics Vidhya*, Analytics Vidhya, 23 Apr. 2018, www.analyticsvidhya.com/blog/2018/04/a-comprehensive-guide-to-understand-and-implement-text-classification-in-python/.

[3] KazAnova. “Sentiment140 Dataset with 1.6 Million Tweets.” *Kaggle*, Kaggle, 13 Sept. 2017, www.kaggle.com/kazanova/sentiment140.

[4] Jain, Shubham. “Ultimate Guide to Deal with Text Data (Using Python) - for Data Scientists & Engineers.” *Analytics Vidhya*, 27 Feb. 2018,

www.analyticsvidhya.com/blog/2018/02/the-different-methods-deal-text-data-predictive-python/.

[5] "Getting Started with the Keras Sequential Model." *Keras Documentation*, Github, keras.io/getting-started/sequential-model-guide/#specifying-the-input-shape.

[6] Kvamme Repp, Oystein. *Event Detection in Social Media*. Norwegian University of Science and Technology, June 2016, brage.bibsys.no/xmlui/bitstream/handle/11250/2410729/14737_FULLTEXT.pdf?sequence=1.

[7] Bahrami, Mohsen, et al. *Twitter Reveals: Using Twitter Analytics to Predict Public Protests*. Massachusetts Institute of Technology, 2016, arxiv.org/pdf/1805.00358.pdf.

[8] Liu, Xiaomo, Li, Quanzhi, Nourbakhsh, Armineh, Fang, Rui, Thomas, Merine, Anderson, Kajsa, Kociuba, Russ, Vedder, Mark, Pomerville, Steve, Wudali, Ramdev, Martin, Robert, Duprey, John, Vachher, Arun, Keenan, William, Shah, Sameena, et al. *Reuters Tracer: A Large Scale System of Detecting & Verifying Real-Time News Events from Twitter.*, 2016, https://www.researchgate.net/publication/309471330_Reuters_Tracer_A_Large_Scale_System_of_Detecting_Verifying_Real-Time_News_Events_from_Twitter.

About the authors:

James Pala is an undergraduate computer science student (Class of '20) with a deep interest in the future of smarter computer systems and intelligence, and how they can be designed to improve the human condition.

Taylor Wong is an undergraduate computer science student (Class of '19) with a curiosity to learn about various neural network applications, as well as how machine learning will be affecting and influencing future generations.

Marcus Rogers is an undergraduate computer science student (Class of '20) with an interest in algorithms and the potential use of computers.

APPENDIX A: SUPPLEMENTARY INFORMATION

Dataset 25 Labels

Label 1	Colorado Wildfires
Label 2	Costa Rica Earthquake
Label 3	Guatemala Earthquake
Label 4	Italy Earthquake
Label 5	Philippine Floods
Label 6	Pablo Typhoon
Label 7	Venezuela refinery
Label 8	Alberta Floods
Label 9	Australia bushfire
Label 10	Bohol Earthquake
Label 11	Boston Bombing
Label 12	Brazil Nightclub fire
Label 13	Colorado floods
Label 14	Glasgow helicopter crash
Label 15	LA shooting
Label 16	LAC train crash
Label 17	Manila floods
Label 18	NY train crash
Label 19	Queensland floods
Label 20	Russia Meteor
Label 21	Sardinia floods
Label 22	Singapore haze
Label 23	Spain train crash
Label 24	Yolanda typhoon
Label 25	West Texas Explosion

Dataset 12 Labels

Label 1	Fires
Label 2	Earthquakes
Label 3	Floods
Label 4	Typhoons
Label 5	Refinery
Label 6	Bombings
Label 7	Helicopter Crashes
Label 8	Shootings
Label 9	Train Crashes
Label 10	Meteors
Label 11	Haze
Label 12	Explosions