Twitter has become an important communication channel in times of emergency. Since nearly everyone has a smartphone, people can "announce" an emergency they’re observing in real-time. Because of this, more agencies are interested in programmatically monitoring Twitter (i.e. disaster relief organizations and news agencies). But, it’s not always clear whether a person’s words are actually announcing a disaster.

Take this example:

This project attempts to predict which tweets are about real disasters and which are not. The dataset is from a Kaggle Machine Learning competition.

Dataset

The overall dataset (corpora) contains 10,000 tweets (documents) of which 7,613 are the training dataset and have been classified (by a human) as "emergency" or "not an emergency". This is known as "supervised model". A further [INSERT NUMBER] are part of a testing dataset for which the classification is unknown to us. Of the training set 4342 tweets are classified as 0 or "not an emergency" and 3271 are classified as 1 or "emergency" - the dataset is not perfectly balanced.

Tweet text examples:

* "13,000 people receive #wildfires evacuation orders in California"
* "Forest fire near La Ronge Sask. Canada"
* "I'm on top of the hill and I can see a fire in the woods..."
* "I'm afraid that the tornado is coming to our area..."
* "Crying out for more! Set me ablaze"
* "What a wonderful day!"
* "#Flood in Bago Myanmar #We arrived Bago"

Methodology

Making predictions such as "real" or "not real" or "good" or "not good" from raw text is known as a classification problem. Taking text and analyzing in a way that a machine learning model can make prediction is call Natural Language Processing (NLP). This project uses the python library scikit-learn to analyze the text and make predictions.

Tokenization

Tokenization converts the training dataset to a list of individual unique tokens (known as the vocabulary - most of which will be words. With no filtering this dataset has 21,637 tokens.

Vectorization

Each document (tweet) is then transformed into a vector with 1 row and 21,637 columns where each column represents a word from the vocabulary. 0s and 1s represent the absence or presence of that particular word. The corpora consist of a 7613x21637 sparse matrix with 111497 stored elements in Compressed Sparse Row format.

Baseline

The problem is now a numeric one and initial logistic regression model can be built. The mean cross-validation accuracy is 0.711. After tuning the result by adjusting the regularization parameter C, the cross-validation accuracy is improved to 0.730.

Improving the model

CountVectorizer extracts tokens using a regular expression "\b\w\w+\b" which finds all sequences of characters that consist of at least 2 letters or numbers (\w) and are separated by word boundaries. It splits contractions like doesn't into doesn and t. It converts all words to lowercase.

Lemmatization is the process of contracting similar words and plurals down to the verb root or lemma. Lemmatization reduces the token count from 21,637 to 14,703.

Increasing the minimum word frequency can remove low-frequency words which are often garbage or misspelled words e.g. '0ubg9wfyge', '0uk0h9hozn', '0usdi5jcho' . An optimization for min\_df was run using values in the range 1 to 50. The best cross-validation accuracy was 0.73 at a min\_df of 4. Although not an improvement the number of features is reduced from 14,703 to 2,795

Rescaling data with tf-idf and n-Grams

Term Frequency (TF) - Inverse Document Frequency (IDF) - i.e. give a high weight to any term that appears often in a particular document, but not in very many documents in the corpus. If a word appears many times in a document, but not in very many documents, it is likely to be descriptive of those documents.

The approach up until now has been to discard the word order and only consider a so-called “bag of words”. Meaning in text also comes from the order of the words and to attempt to simulate this n-Grams which link consecutive words to create new features are created.

The heatmap shows cross-validation scores versus 1-1, 1-2, and 1-3 n-Grams versus the hyper-parameter C.

Conclusion

The model was difficult to improve beyond the initial cross-validation score of 0.73 – even using lemmatization, the td-idf methodology and n-Grams. The number of features were reduced considerably – almost by a factor of 10 and could have gone much lower with only a minor reduction accuracy. On larger datasets this would improve the processing time.

Perhaps the amount of information contained in a tweet is too small and the critical information is contained in single word. The above visualization shows the most significant tokens with their model coefficients.