Recognizing the Real Disaster Tweets

This project targets using Natural Language Processing methods to create models that can identify if a tweet is referring to an ongoing disaster. The subject was decided in conjunction with the data science resource and competition website kaggle and can be seen at https://www.kaggle.com/c/nlp-getting-started/overview.

Many tweets, often through regular use of hyperbole, will use terms associated with disaster scenarios to talk about everyday occurrences. But if you could mechanically discern the real events from the hyperbole, an automated tweet reader could surface those tweets that refer to actual emergency situations, and help direct rescue responses expediently. With the location information often included in a tweet, such tools could even pave the way for generating hot spot maps to start sending crews out before calls referring to incidents are able to be triaged. Further work could also one day automatically update dispatches already en route to such locations, add tweeted pictures to incident reports, and lower the priority on calls that seem to be originating from an already dispatched to area.

The models to automatically identify disaster tweets would therefore be of import to any emergency response dispatch service. They would be able to deliver aid quicker, and perhaps in some cases even deliver aid where they wouldn't have known to at all, thereby saving costs in property damage, and human life/injury. Since the primary consumers of these models would therefore typically be public services that could not afford to budget for these models, Kaggle is stepping in to recognize the value of competent models for this problem space with a site hosted prize pool.

To train the models for this project, the company figure-eight has provided a dataset to Kaggle of a manually tagged set of ~10,000 tweets in easy to import csv format with each text tweet tagged as disaster or not, and a subset of these tweets also having fields with their contained disaster keywords and location information. In order to engage in the spirit of the Kaggle competition, the goal will be to form an accurate trained model for identifying disaster tweets from just these provided tweets. We therefore won't be wrangling any additional data sets to complete our model, but will perform various preprocessing steps to increase the predictive power of our closed data set.

We'll evaluate various models to determine what has the best predictive capability. We will use a TF-IDF representation of our processed input for standard machine learning approaches, as well as modern NLP techniques with neural networks. We will be evaluating the performance of our models locally via cross validation in the training set, and then additionally via kaggles provided test set which must be submitted for actual F1-score validation as, by kaggle's admission, the test set has had some falsified entries introduced to prevent hardcoding of solutions. Accompanying our kaggle submission will be a github hosted report on our overall process, slides to summarize the same, and jupyter notebooks with further details of our process.

Tweet Pre-processing

For this closed Kaggle NLP problem the primary ask in our data gathering step isn't on amalgamating various sources, but rather on transforming our one data set for maximal utility. While we are provided with some additional information beyond the text on some of the tweets in our data set (namely keywords and location) we'll be disregarding these as the primary intent of our project is to form judgments of the threat level of a tweet based on the text alone.

In approaching our dataset then our first step is to ensure we do indeed have entries in both the text field for every tweet and the label for each tweet: disaster related or not related. Having verified this to be the case we can move on to the steps generally taken in preparing a text corpus for machine learning.

There are a few different general methods with which we can transform text to increase the signal to noise ratio. While some are less useful when the entries in the corpus you're examining are each no longer than a tweet, some steps retain their value. Chief among these is first reducing words to their lemmas, so that different conjugations of a word all have matching representations and can be correctly assigned the same import and predictive value, and second removing so called 'stop words.' These words such as 'the', 'I', or 'as' are liable to appear in any text in great frequency regardless of the general meaning of that text. Since the goal in machine learning for natural language processing is typically to predict the intent of texts, words that would appear along with any intent with no clear relation on how often they would appear should be regarded as not having any predictive power.

Since these are usually the first steps for any NLP problem, there exist a variety of libraries that already reduce words to their lemma and collect the stop words that you may want to remove. We use one of the more widely used python libraries for this process: spacy. Spacy is a comprehensive library that can split up a provided text into 'tokens' and classify each token in various ways. Beyond being able to reduce words to their lemma and match from a pregathered collection of stop words, Spacy in fact stores whole vocabularies and can help us sweep typos from our predictive store, and can also identify punctuation that we'll also wish to remove due to low predictive value.

Another of the reasons for widespread Spacy use is the ability to expand on the default text processing it undergoes. This means we can add another prebuilt library onto the base implementation so that it can also identify 'emoticon' tokens that it would otherwise have thrown out as punctuation, but that quite possibly could have predictive value in our tweet context. Additionally in our unique context we'll encounter words used as hashtags, which could be expected to be used in ways other than the words on their own and we'll want to keep this context. We will therefore be checking the 'left of' token on each of our words as we process them and keeping hashtag words as one word in our final processed tweets.

Before keeping our processed tweets after these removals we also collect and output the most common words among all the tweets to see if we can spot some 'stop words' that Spacy did not.

We end up additionally removing the tokens "u", "2”, "4”, "'s", "s", "|", "amp" and “” from our texts as obviously of low predictive value. With a sanity check of our final tweets we also identify that some hex escape characters are now in our corpus where they weren’t before, and add logic to clean these entries. Finally we spot that Spacy has inexplicably added ‘-PRON-’ to many of our processed tweets and remove this as well.

Topic Modeling

It’s said that we can know a word by the company it keeps. For larger texts this means that we can put together words that often appear together in a text, and pull out certain ‘topics’ about a text as a whole if we can identify a group of words in that text. This sort of modeling can give us useful summary insights (in our case it would be great to be able to pull out ‘disaster’ or ‘safe’), and since it’s generated based wholly on counts of words appearing with other words, we can generate these topic models without our program having to know what exactly it is looking for, and can assign topics to generated groupings after the fact. Training such topics however requires we have enough variety of texts to select over however. And while we make an attempt to create such models on our collection of tweets, our topic modeling appears random at best, indicating that the short length of tweets simply does not provide enough information for this task.

TF-IDF

Having already processed much of the noise words out of our tweets, we can also produce one more useful representation for some of our machine learning models. We know that our machine learning models are agnostic to things like the meanings of words, and will need some kind of numerical representation in order to perform. When it comes to natural language, one commonly utilized vector representation is the bag of words.

To produce these, first the sum of unique words across a corpus must be computed. Then for each document in a corpus your bag of words vector is a vector equal to the length of the total vocabulary (this is partly why removing stop words and punctuation is a helpful first step) with each dimension in the vector corresponding to one of the tokens in the vocabulary, and the number at that dimension for a document being the count of that token in the document (tweet in our case). These vectors are often sparse, and will definitely be so in the case of our short tweets.

Our naive ‘bag of words’ representation can however be improved into a better input model with a little extra computation however. When we begin training our machine learning models we will be trying to determine which dimensions are most likely to influence a label of ‘disaster’ or ‘not disaster’. It’s clear that the words that will be most important for this task are words that strictly appear with one label or the other, and indeed the more often a word appears at all, the less likely it is to be able to strictly define a tweet one way or the other, and the more likely it is that it’s an unidentified stop word.

In order to raise the significance of each of our dimensions, we will transform them with a method known as Term- Frequency / Inverse Document Frequency (TF-IDF). With this technique each term is weighted not just with its appearance count (Term Frequency) in a document, but also how many documents it appears in as a whole. The 'Inverse Document Frequency' comes into play with a term being degraded in weight the more documents it appears in. This gives us proportional values for each term that better represent their predictive values, and should produce more reliable models with standard machine learning techniques.

Tweet Exploration

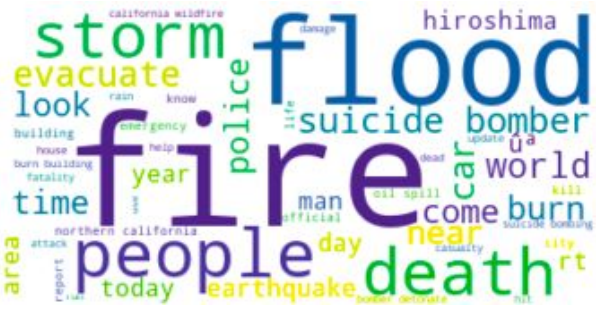
In the world of NLP there are limited exploration steps that we can take to describe our problem space. Indeed everything comes down to how particular words (or sometimes statements) map to particular intents.

Word Clouds

In our case, short tweets mean words are the size of our interest, so we wish to visualize what words align to tweets that are 'neutral' vs those that refer to an ongoing disaster, as identified by the company figure 8 with our data set. Using our already preprocessed set of tweets, we can combine the text of all 'neutral' tweets together, and all 'disaster' tweets together to form 'word cloud' representations, in which we print all of the most commonly appearing words in a graphic visual, but have greater font sizes for words that appear relatively more frequently in the collection. For each data set this yields:

Neutral

Disaster

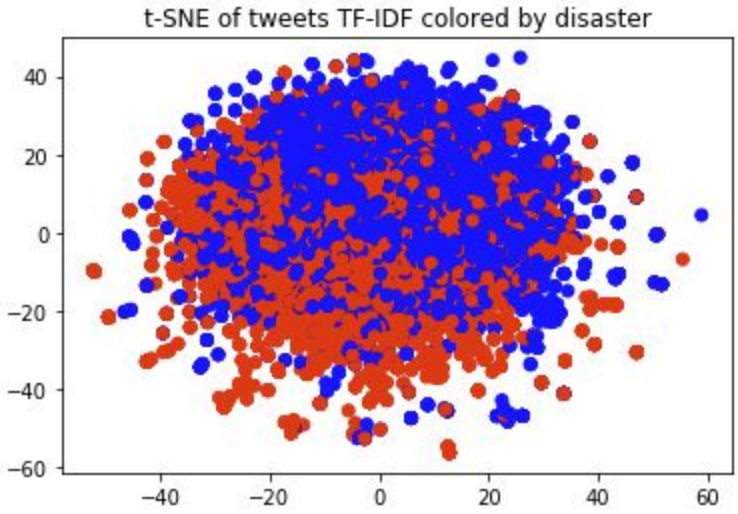


Initial takeaways are that it's odd to see 'body bag' in the neutral collection of tweets, and that there's probably a little more cleanup we want to go back to before passing into our machine learning models. Namely, we first want to remove the non interpretable characters (ûª), and judging influence of tokens by their relative sizes, we want to remove 'rt' across the board which we see being nearly equally represented in both data sets. We revise our preprocessing notebook accordingly and going forward with machine learning will use the tweets with these removed.

SNE

Another method to visualize if there are distincitive differences between two different corpuses (in our case, disaster tweets and non disaster tweets) involves the usage of our TF-IDF

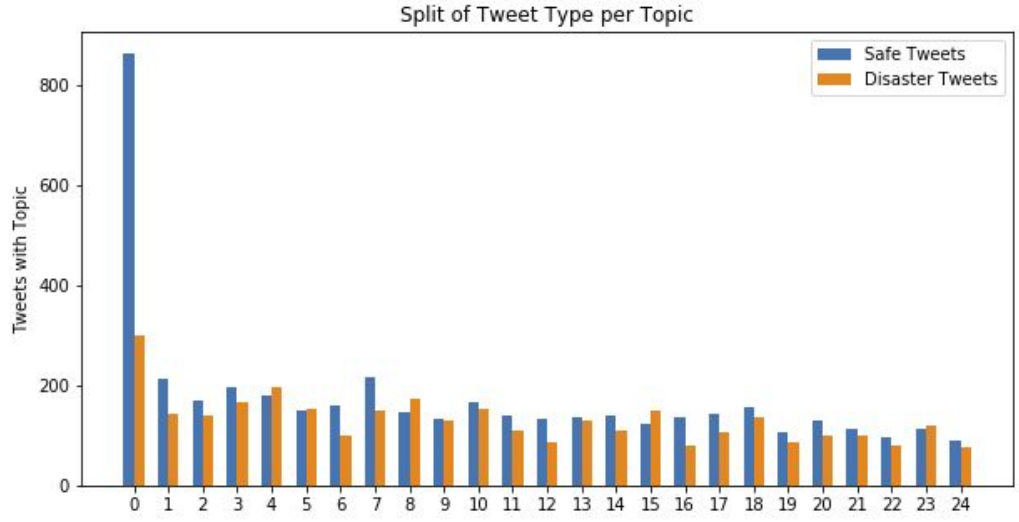
representation of our tweets. Via an algorithm known as SNE we can visualize the high dimensional spaces (our TF-IDF vectors) in just two dimensions, with the algorithm focusing on preserving the effective distance between all points in a collection as it's goal when transforming down to two dimensions. If we color the points in our two different data sets differently then, we may be able to see if there's a difference at a glance between the two corpuses. Applying the algorithm in this way arrives at, and coloring 'safe' tweets blues and 'disaster' tweets red arrives at



And while it's not a distinct separation, we can spot a some separation about the diagonal from top left to lower right, which gives us some hope for the separability of our two classes in machine learning.

LDA

Finally we turn to our LDA topic modeling. We recall that we didn't have much faith in the usability of this algorithm due to the short length of the documents in our corpus. We can still view counts of each tweet in each of our modeled topics however, and doing so arrives at:



which confirms that no topic is distinctly ‘safe’ or ‘disaster.’ Topic 0 might earn our curiosity though due to the difference in representation, but looking at the words gathered in this topic shows us it’s not exactly a ‘safe’ set:

- like
- survive
- suicide
- wreck
- day
- kill
- think
- war
- traumatised
- good
- police
- let
- come
- bombing
- bad
- car
- stop
- crash
- kid
- zone
- look
- need
- 70
- watch
- survivor

So we conclude these topics are not useful for the machine analysis we will be turning to next.