Analyzing Tweets for Topic Classification

Logan Short, Vishnu Sundaresan, Christopher Wong
Department of Computer Science, Stanford University
{lshort,vishnu,crwong}@stanford.edu

Abstract

(TODO)

1 Introduction

With an overarching goal of making sense of the massive amount of Twitter data in the form of Tweets, our project focuses on characterizing Tweets by their associated specific topics, ignoring the occurrences of hashtags. In addition to simply using standalone learning models such as a bag-of-words model, neural networks on the raw frequencies, or similarity measures, we create an ensemble out of many features of a Tweet. By using the information provided by other information extraction methods such as sentiment analysis, our learning models will potentially be able to yield better results and handle more robust instances of Tweets. This project has the additional utility of being able to be integrated into other supervised learning models that require further dimensions and information about raw Tweets. The generated mapping from Tweet content to hashtag could also be used as the basis for a hashtag recommendation system for Twitter.

2 Previous Work

We first surveyed recent literature and research into analyzing Twitter data to see get an idea of what has already been done. Sections 2.1 to 2.3 provide a brief of overview of 3 papers which significantly influenced and motivated our project. Section 2.4 outlines the new ideas and goals behind our work.

2.1 Lee, Palsetia, Narayanan, Patwary, Agarwal, Choudhary

The basis for Lee et. al. and its motivation is related to Twitter's Trending Topics product, which appears at the stream homepage for users. Most of the time, it is hard to classify or understand what these topics are really about, so the authors set out to classify these topics into 18 different general categories. Lee et al. [1] used two different approaches to solve this problem, including the Bag-of-Words model as well as a network-based classification. In order to come up with the labels initially used for training, they used up to 3 human annotators, with the third being used in the event that the first two did not agree on a category. Focusing on the Bag-of-Words model, the paper constructed word vectors from the trending topic definitions and related Tweets and used tf-idf weighting with a multinomial Naive Bayes classifier to classify topics that appeared. We plan on using some of these same techniques in our implementation of the topic classifier.

2.2 Ramage, Dumais, Liebling

Ramage et. al [2] also discuss two different features that Twitter and it's users would find useful: the process of user and figure discovery, as well as feed filtering based on ones interests. Their work revolves around the use of a partially supervised learning model, Labeled LDA, in order to process the multi-labeled fields in Twitter data and separate it into different dimensions. The concept of labeling each document, or Tweet in this case, by a specific set of topics helps to characterize and better understand the following and reading behaviors of its users. The four main behavior classes that the paper considers are substance, status, style, and a social context to each Tweet, with data that does not fall into one of these categories being separated. Some interesting observations in specific were that certain Twitter features such as mentions, hashtagging, emoticons, replies, retweets, and favorites all were related to specific behavioral characteristics. Our project would help augment this approach by coming up with a way to label Twitter stream data, as well as potentially take away from their conclusions in our own topic classification.

2.3 Li, Ritter, Cardie, Hovy

The process of extracting events and classifying them based on Tweets by Li et. al. [3] can be broken down into three separate areas of work including user-level analysis, identification and extraction of public events, and learning from these areas to acquire more data. The paper proposes a pipeline by which these different areas come together to be able to classify a users Tweets into specific life events. Their process first takes a noisy input stream of user Tweets, and attempts to divide them up into different categories of major life events (many are not seen or captured). In order to better recognize when a major life event is captured in a Tweet, they leverage Tweet replies to determine when congratulations or condolences were being exchanged. Next, Li et al. took these different categories and trained a classifier that would use words in the Tweet, named entity tags, and the top words associated with each pre-determined category to determine which life-event (if any) it belonged to. Different signals such as the sentiment of a Tweet as described by Li et. al. can prove to be very useful, a strategy we plan on employing rather than using simply the raw text for training.

2.4 Motivation and Goal for our Work

Among a large number of the papers we reviewed, we found a consistent trend in that each group applied different approaches to the same problem. The majority of papers begin with a first attempt to solve the problem based purely on an information retrieval-esque language model such as n-gram likelihoods, Bag-of-Words representations, or similarity measures. These often came to the conclusion that many aspects or features in the data were being largely unused, or could not be feasibly used. The next attempts to make sense of the data involved supervised learning in the form of a neural network, or trying to uncover relationships between features using methods such as SVM. Some papers recognize the benefits of using each of these techniques, and create ensembles of different signals and successful methods in order to improve upon their accuracy.

The goal of our project is to create a simple Tweet classification system that predicts the topic of Tweet given its text content. We aim to model Twitter data using various natural language understanding (NLU) techniques similar to those covered above and also methods discussed in CS 224U. One of the overarching takeaways from the previous work is that ensemble systems usually yield improved performance over standalone models and learning algorithms. Previous ensemble algorithms, however, seem to use only models that are all specific to one general area of NLU, such as focusing on word representations or semantic parsing. Our system uses models based off of different areas of NLU, and in Section 5.2, we implement a new, data-driven ensemble algorithm that attempts to yield the best performance.

3 Data Collection and Preprocessing

We begin by discussing our data collection in Section 3.1, and in Section 3.2, we describe our methodology for cleaning and preparing the dataset of Tweets.

3.1 Data Collection

Twitter contains a wealth of data on an extremely diverse set of topics, and it has been often been used as a primary source of data for natural language understanding and machine learning research. Twitter, as a company, has also enabled open access to their data through company-supported APIs. As such, Twitter data is relatively easy to find and many sources of high quality Twitter data have been made available online. Our data set, provided by (TODO) STANFORD-GROUPHERE, contains approximately 8 gigabytes of documents containing Tweet content and metadata. The data is divided into 27 groups based on the presence of certain hashtags.[1] The overall choice of groups provides a nice amount of variation in content. For our system, we used the identifying hashtag as the topic label.

Given the scope of our project and the hardware available to us, we chose to look primarily at a smaller portion of the data containing 6,000 Tweets from each of the 27 topic groups. This yields a total of 162,000 total Tweets. This smaller subset of the raw data allowed us to work more flexibly and efficiently. To verify the appropriateness of our partition size, we initially generated

[1]The complete list of groups (with the # symbol removed) is: `android`, `basic`, `coffee`, `dontjudgeme`, `earthquake`, `egypt`, `election`, `freedom`, `god`, `haiti`, `happy`, `harrypotter`, `healthcare`, `immigration`, `indonesia`, `ipod`, `love`, `mubarak`, `obama`, `obamacare`, `question`, `sotu`, `teaparty`, `tsunami`, `usa`, `win`, `wiunion`.

some of our intermediate results on larger subsets of the entire dataset. The results were sufficiently close enough to indicate that our primary working set was indeed large enough to capture the bulk of language-based variance among the Tweets. From this point on in our paper, any mention of the "dataset" are referencing this subset.

3.2 Tweet Tokenization

The vast majority of content on Twitter is generated by users, and often times, Tweets do not follow formal language conventions. As a result, it is a common occurrence for Tweets to contain words or tokens that are not normally considered to be standard forms of language. Developing a system to accurately parse and extract informative tokens from "noisy" Tweets thus becomes integral to extracting a suitable language corpus and can have a significant impact on the performance of language based analysis of Twitter data.

Our system for the tokenization of Tweets draws heavily from the Twitter preprocessing methods laid out by Pennington et al. in [4]. Several non-standard language tokens which are commonly found in Tweets are filtered or modified during the tokenization process. The first of these are the hashtags themselves; formally, these are tokens which containing a phrase that is preceded by a # symbol, such as "#win". For the purposes of our classification system, hashtags are removed from the Tweet body during preprocessing. This is done because our topic groups are divided based on the presence of certain hashtags, as described in Section 3.1. Our system also modifies emotion faces (emoticons) such as ":)" and encodes them with unique identifier tags. Furthermore, words typed out using only capital letters, such as "NEWS", and words spelled using unnecessary repeated instances of letters, such as "heyyyyy", are labeled using similar tags. Finally, all numerical values are also replaced with a numerical value tag, although URLs and references to other Twitter users (handles, which are denoted by the @ symbol) are left as they appear in the original Tweet. Finally, standard punctuation symbols such as periods and exclamation points are pruned from the Tweet context.

Following the initial processing of the Tweet body, a list of tokens is then obtained by simply splitting the Tweet on whitespace. The flexibility of this tokenization method allows for a high level of context information extraction even from Tweets containing a large amount of unorthodox language. For example, (TODO) NICE EXAMPLE.

4 Learning Models and Features

We now discuss the technical details of the models and features used in the implementation of our system. Our project was primarily coded in Python with the assistance of the popular open source `scikit-learn` module. In Section 4.1, we discuss the construction of our Word-Tweet matrix, a data structure analogous to a general word-document matrix. In Sections 4.2 to 4.6, we discuss our various approaches to modeling Tweets in our data set.

4.1 Word-Tweet Matrix

The first step in our Tweet analysis procedure is the construction of a Word-Tweet matrix. In order to construct such a matrix, the Tweet tokenization process described in Section 3.2 was first applied to all Tweets contained in the dataset. Then, we counted the number of times each unique word token appeared in our dataset. We chose to remove all tokens that did not appear at least 50 times, since rare words are not likely to be helpful in generating accurate models. The remaining frequently-occurring tokens are then placed into a term set and each token is assigned a unique id. A term frequency vector t of a Tweet is then defined to be a vector containing a feature for each of the terms or tokens in the frequently occurring term set. The ith element of the vector, t_i, is equivalent to the number of times the term with id i appears in the Tweet.

The Word-Tweet matrix is then constructed such that the term frequency vector of each Tweet in the dataset is stored as a column in the matrix. This is done using a second pass over the dataset whereby the term frequency vectors of each Tweet are calculated and stored. Any Tweets which have over 50% of their tokens not appearing in the generated term set are not placed into the Word-Tweet matrix as these Tweets have had a significant amount of their content stripped and thus cannot be classified meaningfully. The Word-Tweet matrix is a very useful data structure that provided the basis for our various models, such as raw term frequency vectors and term frequency-inverse document frequency (tf-idf) vectors.

4.2 Raw Term Frequency Vectors

The first classification component we used revolved around the construction of several simple classifiers that classified directly on a bag of words language model in which the feature vector of each Tweet simply contained the counts of the tokens appearing in the Tweet. Term frequency vectors for each Tweet were obtained by normalizing the columns of the Word-Tweet matrix generated using Section 4.1. These frequency vectors were then mapped to their associated hashtag and input as data points for the classification training of our models. Using the raw term frequency vector model as a backbone, we focused on three classification algorithms: logistic regression, k-nearest neighbors, and a 3-layer nerual network.

4.3 Term Frequency-Inverse Document Frequency Vectors

Another possible method for accurately classifying Tweet hashtags involves first constructing term frequency-inverse document frequency or tf-idf based word vectors that encode information about the similarities and differences between different tokens in the corpus. In the tf-idf word vector model, the idf of a token t is given by the formula:

$$idf(t) = \log\left(\frac{\text{\# of total documents}}{\text{\# of documents containing } t}\right)$$

The tf-idf value for a token t and a Tweet d is defined as:

$$tf - idf(t,d) = tf(t,d) \times idf(t)$$

Here $tf(t,d)$ represents the number of occurences of t in the Tweet d. Calculating the tf-idf scores for each of the token-Tweet pairs in the dataset allows us to The tf of the same token t is given by the number of occurences of t in a given Tweet d. The tf-idf value for a Once such vectors have been constructed, Tweets can be represented as a combination of the word vectors for the tokens contained in the Tweet. The Word-Tweet matrix constructed in Section 4.1 allows for straightforward computation of tf-idf based word vectors. Vector representations of Tweets are then constructed by fidning the mean of the word vectors of each word appearing in the Tweet. The resulting vector representations are then used for classification.

4.4 Feature Vectors Derived from GloVe Word Representations

Another approach we tested was building feature vectors for each Tweet by leveraging GloVe word representations. The constructed Word-Tweet matrix allows us to quickly extract the most significant words and their frequencies from each Tweet, and most of these words have a corresponding GloVe vector representation. Then, to build a feature vector for a particular Tweet, we tried many different approaches to combine the GloVe vectors of the significant words in that Tweet. The most straightforward approach was to add all of the vectors of the words together (weighted by the words frequency in that Tweet), and then average the resulting vector sum.

4.5 Sentiment Learning and Scoring

One signal that involves information extraction and additional training on a separate Tweet dataset is sentiment. The predicted sentiment of a particular Tweet could potentially be very useful for identifying the class of certain topics in our dataset by providing a signal that could have its weight in the classification determined by each learning model. For example the win topic should have a very strong correlation towards positive sentiment, and the tsunami topic should be very negative in general.

In order to extract this information to be input when operating on our test dataset, we first trained a sentiment scorer on the Sentiment140 Twitter in order to provide a classification of the sentiment on our topic dataset. This was done by creating a tf-idf feature vector as before on the training data and using the sentiment score labels as the target, and using this learned model to create a pipelined process that generates a sentiment label when training and testing on the topic dataset. The initial training resulted in a test accuracy of 0.712 using logistic regression, and up to 0.739 when using a neural network, which was high enough for us to incorporate into our topic classification model as a training signal.

This semi-supervised strategy might incorrectly label certain Tweets sentiment, but our hope was that for the much larger proportion of Tweets the labeling would be accurate, and create a useful signal for our model to learn. This resulted in an overall test accuracy on the topic dataset of 0.

Model	Precision
Baseline (Guessing)	0.5
Logistic Regression	0.712
3-Layer NN	0.739
kNN - Minkowski	0.672

Table 1: Sentiment classification accuracy.

4.6 OpenIE KnowItAll Relation Extraction and Weighting

KnowItAll is a information extraction open-source tool that takes a sentence and breaks it up into relational clauses. We saw this as a potentially useful addition to our learning models, which would help to differentiate between the importance of specific words in the Tweet, or allow us to exploit the relationship between words. Each n-ary extraction will put emphases on certain words or phrases within the Tweet, allowing us to come up with an intuitive weighting scheme that promotes words in our respective word vector representations. For example take the text The U.S. president Barack Obama gave his speech on Tuesday to thousands of people. The system will create the relation (Barack Obama, is the president of, the US) among others, allowing us to use this information and potentially weight the subject and relational terms more than the rest of the phrase.

Rather than creating a new feature for each of these different relations, we chose to weight the word representations instead based on the most likely relation as outputted by KnowItAll. This decision helps eliminate the problem of having extremely sparse training and test data in our feature vectors as well as allow for human intuition to play a role in determining what relations could potentially be important.

5 Learning Algorithms and Results

We now discuss the results yielded by training different learning algorithms on our various models. Individual model results and analysis are given in Section 5.1, while Section 5.2 discusses our overall ensemble system.

5.1 Individual Model Results

As discussed in Section 4, we modeled our data set using a variety of different approaches: raw term frequency vectors (which we will refer to as FREQ), raw tf-idf vectors (TFIDF), feature vectors derived from GloVe word representations (GLOVE), tf-idf vectors with an additional sentiment scoring feature (SENT), and tf-idf vectors based off of relation extraction weighting (REL).

We trained on each of these models using three common learning algorithms: logistic regression, k-nearest neighbors classifier using Minkowski distance, and a 3-layer shallow neural network. The neural network implementation was adapted from the CS 224U `distributedwordreps` code lab. For testing data, we used approximately 500 additional Tweets from each topic group that were not included in the original subset based on 6000 from each group. Each classifier outputted the predicted topic label for each Tweet in the test data set, and test accuracy was measured by simply calculating the percentage of test Tweets that were correctly classified. Table 5.1 summarizes the results.

(TODO) Analysis. What do we see. Error analysis. Why do things like sentiment and relation extraction not work too well individually? Give examples of where sentiment and relation extraction fail. Give examples of Tweets in general that were difficult to classify. GloVe vectors not too strong.

5.2 Ensemble Model

As mentioned in Section 2.4, previous literature suggests that ensemble systems can potentially improve performance. After analyzing the strengths and weaknesses of each model in Section 5.1, we predicted that an ensemble system which incorporated different areas of NLU could potentially work very well. Since the models described above use some very different features, the strengths of each could make up for other models' weaknesses.

To start our implementation, we focused on the logistic regression output of each model. Logistic regression outputs a probability score for each possible candidate label, and the predicted classification (when standalone) is simply the label with the highest probability score. Our ensemble system aggregates the probability scores of all candidate topics for each Tweet. We used a data-driven approach to find the best set of weights possible; our algorithm methodically tests various combinations of weights on each model to find the set that provides the best performance. Thus, the aggregation is a sum of weighted probability scores. For Tweet t, let $\textsc{Model}(t)$ denote the probability scores given by training logistic regression on $\textsc{Model}$ and predicting the label for t.

$$
\begin{aligned}
\textsc{Ensemble}(t) = \alpha \times \textsc{Freq}(t) + & \quad (1)\\
\beta \times \textsc{Tfidf}(t) + & \quad (2)\\
\gamma \times \textsc{Glove}(t) + & \quad (3)\\
\delta \times \textsc{Sent}(t) + & \quad (4)\\
\epsilon \times \textsc{Rel}(t) & \quad (5)
\end{aligned}
$$

6 Conclusion

(TODO) Conclusion

6.1 Future Work

(TODO) Future Work. Different ways to incorporate sentiment analysis and relation extraction. Larger data set. Different topics (ones that work better or worse with sentiment analysis). Hashtag recommendation system.

References

[1] K. Lee, D. Palsetia, R. Narayanan, M. Patwary, A. Agrawal, A. Choudhary. Twitter Trending Topic Classification. IEEE, 2011.

[2] D. Ramage, S. Dumais, D. Liebling. Characterizing Microblogs with Topic Models. AAAI, 2010.

[3] J. Li, A. Ritter, C. Cardie, and E. Hovy. Major Life Event Extraction from Twitter Based on Congratulations/Condolences Speech Acts. EMNLP, 2014.

[4] J. Pennington, R. Socher, C. Manning. GloVe: Global Vectors for Word Representation. EMNLP, 2014.

Algorithm 1 Ensemble system.

```
1: $\alpha, \beta, \gamma, \delta, \epsilon \leftarrow 0$
2: for all Tweets $t$ do
3:     Compute FREQ$(t)$, TFIDF$(t)$, GLOVE$(t)$, SENT$(t)$, and REL$(t)$
4: end for
5: for all $\alpha$ from 0 to 1, every time increment by $0.2$ do
6:     for all $\beta$ from 0 to 1, every time increment by $0.2$ do
7:         for all $\gamma$ from 0 to 1, every time increment by $0.2$ do
8:             for all $\delta$ from 0 to 1, every time increment by $0.2$ do
9:                 for all $\epsilon$ from 0 to 1, every time increment by $0.2$ do
10:                    Compute ENSEMBLE$(t)$ for all Tweets $t$
11:                    Evaluate classification accuracy of $(\alpha, \beta, \gamma, \delta, \epsilon)$
12:                end for
13:            end for
14:        end for
15:    end for
16: end for
17: return Best $(\alpha, \beta, \gamma, \delta, \epsilon)$
```

Model	FREQ	TFIDF	GLOVE	SENT	REL
Logistic Regression	0.58507	0.61264	0.48970	0.56820	0.58764
3-Layer Neural Net.	x	x	x	x	x
kNN - Minkowski	x	x	x	x	x

Table 2: Classification Accuracy for Individual Models