

Predicting Prevalence of Influenza-Like Illness From Tweets ...

Alma Niu

Computer Science,

Santa Clara University, 2018

aniu@scu.edu

Jonathan Pak

Computer Science,

Santa Clara University, 2018

jpak@scu.edu

Haruto Nakai

Computer Science

Santa Clara University, 2018

hnakai@scu.edu

Abstract - This paper focuses on using two classification algorithms, Naive Bayes and logistic regression, to identify whether a tweet collected from Twitter contains influenza-like illness related contents. First, the algorithm is fed a set of around 5,000 tweets containing the word “flu”, tagged as either related or unrelated, collected by the University of Colorado Boulder. Then, extensive experiments, such as 5-fold cross validation, are run to determine the performance of our algorithm. The tests have shown that both algorithms run at about 75% accuracy, with Naive Bayes having slightly better results than logistic regression. Then we used the trained algorithm against sets of recent tweets searched by sickness-related words, such as “flu”, “sick”, and “fever”. Although we had accurate results for those with the word “flu” just like the training dataset, other queries yielded absurdly high ratios of false positives that can be attributed to the mismatch of the ranges of the training data and queries.

1. Introduction

Problem Background-Influenza has been a common season illness that plagues areas around the world, spreading

faster than people have anticipated. Epidemics of seasonal influenza are major health concerns, so public health data and surveillances help government raise awareness, which would lead to better preparation, and hopefully a decrease in infection rates. However, the process of acquiring public health data manually is costly, slow, and laborious. Analyzing data collected from Twitter, a popular social network platform, helps automatically access the spread of flu for a smaller overhead cost. Results from analyzing Twitter data has been experimented in the past and has been recognized to serve as a reasonable proxy for alerting disease outbreak and has a strong linear correlation with real world influenza data. Using the data from Twitter comes with several challenges. Health related tweets are relatively scarce. Also, not all tweets that contain words that could pertain to sickness are actually relevant, since some sickness-related words have multiple meanings. Because we will still obtain these tweets when using search queries to collect data, data scientists must have a way to distinguish between what they want and do not want, preferably done automatically through the use of algorithms.

Questions- Initially, our goal was to predict the trend of influenza outbreak using sick tweets submitted in December 2017 within the bounding region boxing California and Arizona, as flu season peaks around the fall. However, the plan was stifled when we learned that the publicly accessible Twitter live streaming API can only be used to obtain tweets from the last seven days, limiting our effectiveness on tweet status timeline. There was also the fact that most tweets did not have accurate geotags for us to map. Our direction for our experiment shifted to using pre-tagged data to train our algorithm to determine and compare the accuracy of two text sorting algorithms, Naive Bayes and logistic regression. During the course of our analysis, we also wanted to compare the accuracy of our algorithm before and after stripping the training data of stop words, which are irrelevant when trying to determine the meaning of a sentence. After training our algorithms, we took tweets from the last 7 days that contained sickness-related words ran them through the algorithm.

Contributions - In this work, we have 5 main contribution contents:

- (1) We retrieved pre-tagged tweets from Twitter and loaded them onto csv files.
- (2) We used scikit-learn to split our datasets into training and testing data.
- (3) We trained our classification algorithm to learn the contexts of the data.
- (4) We used visualizations and experimentations, such as confusion matrices, recall and precision, and 5-fold cross validation, to analyze the effectiveness of our classification algorithm.
- (5) We applied our algorithm to real time tweets

2. Related Works

Twitter data has been used by different researches to determine prevalence of sickness since the popularization of the website. In 2017, Kewei Zhang, Reza Arablouei, and Raja Jurdak from the University of Queensland, Australia, conducted a research on detecting spread and distribution of influenza-like illnesses in the continent-nation of Australia using a sample of 8.9 million geotagged tweets collected in Australia, showing that there is a strong correlation between the number of influenza-related tweets and the reported influenza victims in the certain area, and that Twitter data can be useful in detecting outbreak of diseases in Australia. We started this project as an application of their methods in a smaller scale in the San Francisco Bay Area, with plans to take not only the experimental samples, but also the training data sample from the Twitter stream, and map the occurrences of influenza-related tweets. We have transitioned to the tests we have right now due to time constraints and the

barrier of entry to using enterprise-level Twitter API privileges.

Inspired by Professor Manna, we decided to use 5-fold cross validation on both logistic regression and Naive Bayes to analyze their accuracy. Then, we graphed the results to visually see if one algorithm outperforms the other in some consistent pattern.

3. Methodology

In this section, we will briefly explain our algorithm approach and assumptions.

Data Collection - For our training data, we used data collected by Michael Paul of University of Colorado Boulder, under “Twitter Flu Annotations”. Of the several data sets offered by him, we chose to use the datasets that differentiated between influenza-related and non-related tweets. All of the tweets include are from either 2009 or 2012. Each line of his text documents included a tweet ID that points to a specific tweet and an integer, 0 or 1, to indicate its relatedness. In order to process the tweets, we used Python’s twitter-python package to extract the text of the tweet, utilizing the GetStatus function in particular. About a third of the tweets from these data sets were not available to be retrieved, due to permissions (tweets can be hidden or only available to friends), deletion of tweet by the account owner, and suspension of accounts. After these omissions, we ended up with 1581 tweets from 2009 and 3235 tweets from 2012, for a total of

4816 tweets in our training data set. All of these tweets have the word “flu” in the text, whether or not they are related.

In addition to extracting Michael Paul’s data for use as training data, we also collected tweets for us to use the trained sorting algorithm on. We used the twitter-python package’s GetSearch function to retrieve our datasets. We extracted 6 sets of search results, with flu-related words as keywords (“flu”, “sick”, “cold”, “cough”, “fever”, and “virus” respectively), all located roughly within the San Francisco Bay Area, calculated as a circle with radius length of 100 miles with Oakland as the center. The tweets are the most recent tweets with the keyword in the area, with a limit of 5000 data points, all sent within a seven-day range before March 12th, 2018.

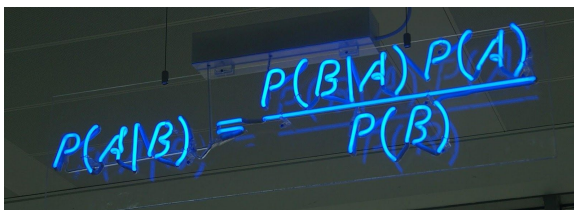
Data Table - In our training dataset of 4816 texts, we have three columns: *Tweet ID*, the unique identification number given to each status update, *text*, the body of the tweet, and *related*, a number of either 0 or 1. Under related, label 1 represents an ill-related tweet whereas label 0 represents a non-ill status.

Training & Testing - We used libraries from scikit-learn to execute our training phase, running our code via Jupyter Notebook running Python. We loaded the training tweets from the csv files onto Pandas dataframes. Then, we called for the train_test_split method to divide our data into various sets: *X_tweet*, *y_tweet*,

X_test,y_test. After that, we fit our X_tweet with CountVectorizer() to transform X_tweet and X_test into a document term matrix. We needed these document term matrices to test our accuracy with Naive Bayes and Logistic Regression

In our Naive Bayes, we fit out data using the X_tweet document term matrix we found and our y_tweet that we acquired through the train_test_split. By fitting our data, we were able to create a prediction which we used along with our y_test data to compute our accuracy through scikit-learn metrics. To test our accuracy with Logistic Regression, we took a similar approach as the Naive Bayes. We had to fit our X_tweet document term matrix and y_tweet through Logistic Regression. After fitting our data, we retrieved our prediction and were able to obtain an accuracy for our Logistic Regression.

Algorithm Description-One of our algorithm is Naive Bayes, a generative machine learning algorithm that uses learned conditional probabilities(Bayes Rule) to calculate the probability that a particular feature set belongs to class.


$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

If predicting a test data, we calculate the probability of the test data for each class.

Then, the test data will belong to the class document with the highest probability. In this method, it is easy to model the tokens(words) in a document.

Another algorithm is Logistic Regression, which uses a linear decision boundary as a means of classifying a set of features and is a discriminative classifier in contrast to Naive Bayes. Logistic Regression covers the case of binary dependent variables, while our project classification is similar to Naive Baye, independent upon categories. Therefore, I assume our Naive Bayes algorithm will have better performance than Logistic Regression. Another assumption we have is that our accuracy will increase after removing stop words for both algorithms,because we got rid of unwanted contents or noise in our data.

4. Experiment And Analysis

A. Data

We tried creating our own program to retrieve current tweets but ran into difficulties causing us to scrap that idea. We later found data collected by Michael Paul of University of Colorado Boulder, under “Twitter Flu Annotations”. This is a selected portion of training data that is annotated. The training data was gathered between 2009-2012 and came with the Twitter ID and an integer,0 or 1, to indicate sickness related or not.

The training data consisted of around 10,000 tweets but only came with the Twitter ID. We had to use the Twitter

API to access the actual tweets however we were only able to obtain 4816 tweets with texts. Although the tweets were marked, it didn't necessarily mean that those tweets were sick related. There are many sick related words that have various meanings that could be unrelated to our search. We created an example vocabulary to see that our algorithms were looking at ILI related tweets.

B. Experimental Setup

We trained the data that we collected through CrossValidation to create our train and text data. Before we could test out data with Naïve Bayes and Logistic Regression, we used the CountVectorizer to fit and transform our training and test data. We also acquired specific data by using a keyword to filter out that tweet to see the accuracy for that keyword specifically.

After loading our retrivedTexts2.csv file , we decided to determine the value of certain words in our text ,using the bag of words method. In our context, the TF-IDF(Term Frequency -Inverse Document Frequency) is a measure of how common a word is in relative to all the words in text. Rare words in TF-IDF have higher value than common words. The formula for TF-IDF is showcased below.

$$w_{i,j} = tf_{i,j} \times \log \left(\frac{N}{df_i} \right)$$

TF-IDF= (number of times each word appears in comment) x log(number of comments in training data/number of comments with word)

Since our data is majority wise ,54% percent , sick tweet related, we can use TF-IDF method to find 6 common 'sick words' in our data. The common words we found are:flu, sick, cold, cough, fever, and virus. These words have lower value than unique words, such as "sun" or "gym" in our text. Then, using the 6 common words, we call twitter API to grab the tweets for each of the common word, creating 6 additional datasets.Then, we will run each of our additional datasets through our Naive Bayes algorithm, evaluating our accuracy on current live tweets.

Figure: TF-IDF Word Results

COMMON

RARE

Sick 3.25	Sun 8.47
Cold 4.49	Beach 7.78
Cough 5.53	Math 8.47
Fever 5.65	Gym 7.78
Virus 4.71	Strong 7.79
Flu 0.1 (very common)	Tree 8.84

From the results above, we can see that the group of common words shows a low value general trend, inferring that our

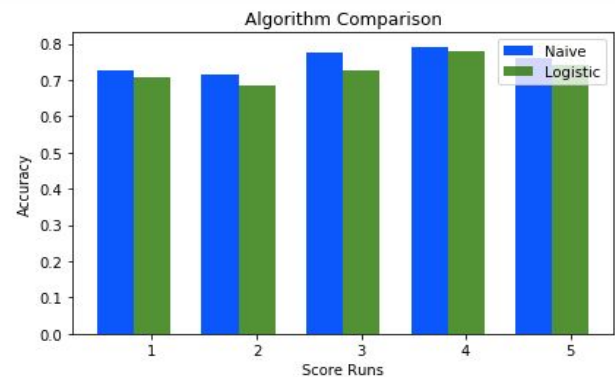
data is mainly sick-related. However, Bag of Words is generally used for categorizing documents. Yet we used it to detect common 'sick' words to further grab live Tweets and test our algorithm.

C. Results

When looking for the accuracy of our Naive Bayes and Logistic Regression test, we approached this in three ways. The three different methods we tested were: 1500 tweets with stop words, 5000 tweets with stop words and 5000 tweets when removing stop words. When running the first test of 1500 tweets, we obtained around a 76.5% accuracy for Naive Bayes and a 74.8% for Logistic Regression. We saw the accuracy decreasing when we added more tweets to around a total count of 5000 tweets. When running the accuracy tests again, Naive Bayes had an accuracy of 74.4% and Logistic Regression had an accuracy of 72.4%. When we saw that our accuracy decrease, we were thinking of ideas that could potentially increase and ended up testing stop words as well. Running the accuracy test again while removing the stop words increased our accuracy slightly. We obtained an accuracy of 74.8% for Naive Bayes and an accuracy of 74.1% for the Logistic Regression. The two tests always resulted in an accuracy close together so we can assume that the code is running properly.

To evaluate both our algorithm, we used 5-fold cross validation. We break our main dataset into 5 equal portions,

testing the algorithm five times on each portion while training the first remaining portions each time. Attached below is our graph.



From the result above, we found that Naive Bayes consistently outperformed Logistic Regression, confirming our initial assumption. Naive Bayes is mostly used in text classification and satisfy the independence rule, having higher success rate as compared to other algorithms.

Another way of evaluating our algorithm is Recall, Precision, and F measure. The results are shown below.

```
('Recall', 0.8135593220338984)
('Precision', 0.7353760445682451)
('F Score', 0.7724945135332846)
```

Recall shows how solid our algorithm detects sick related tweets. Precision shows the probability of how likely a sick tweet will be correctly identified. F Score is the mean of Recall and Precision, summarizing both factors.

Table 1: Keywords and Predicted Relatedness		
Keyword	# of tweets collected	Percentage of “related” tweets
flu	1228	70.358%
sick	4733	94.084%
cold	4893	91.947%
cough	633	93.049%
fever	1780	94.435%
virus	768	45.833%

Table 1 lists our results when running each of the searched data sets through our trained algorithm. We can see that while the results for the word “flu” yielded results closer to our cross-testing (70.358% and 74.169% respectively), the other search queries yielded a much different result. The accuracy of the results with “flu” can be attributed to the dataset’s closeness to the algorithm’s training data; both of the sets are collections of random tweets with the word “flu” in their text. The problem arises in the other datasets, where the keywords can be used in other connotations that does not relate to influenza-like sickness. A large portion of the data is sorted as related despite many of the texts being unrelated to sickness. For example, the term “sick” can be used

as an adjective meaning “disgusting” or “vile”, and is often used by the tweets in the dataset to comment on political figures, notably used against Donald Trump, the sitting US Republican president, collecting hate from the largely liberal California population in the past two years. Vast portions of the data collected used the word in this way. It is also used as a slang in a similar meaning as “cool”, used a few times in the dataset commenting on songs and artists. The other keywords also have alternative uses that are popular among the collected tweets.

The fact that the algorithm showed accurate results on the first data set and did so poorly to sort unrelated tweets as unrelated for the others could be that the trained algorithm is only qualified to sort inputs that are similar in nature to the items in the dataset it was trained with. The nature of the algorithm makes it so that if the input contains vocabulary that was not present in the training dataset, those words would be ignored. With many unrelated terms not included in the training vocabulary, the algorithm only has a few words per tweet to determine if it is related or not after the trim, leading to the large amount of false positives. For example, we have the text “RT @McFaul: This is sick. Remember, Kiselev works for Putin.”, a tweet that was captured in the search for the search result for the word “sick” 98 times. Removing stop words (“This”, “is”, “for”) and proper nouns irrelevant in

2009 and 2012 contexts leaves only the words “sick” and “remember” left in the text for the algorithm to figure out whether or not the tweet is related to influenza or not. Because all tweets including the words “sick” and “flu” (search query for the training data) probably has to do with sickness, the above tweet will also be sorted into the related category, even though a text has no relation and a lot of irrelevant words. If we were to have a much larger set of training data that had been tagged, which includes tweets similar to those we see in the datasets we tested the algorithm on, the algorithm would have been able to take them into consideration. For example, in the tweet above, if other tweets that have to do with politics had been tagged as unrelated, the algorithm would recognize the words such as “Kiselev” and “Putin” as words unrelated to influenza rather than simply ignoring them.

4. Conclusion - Our experimentation found that classification of tweets with the word ‘flu’ in its text into groups of which one is related to influenza-related illnesses and the other is not using Naive Bayes algorithm and logistic regression, resulting in 76.5% and 74.8% accuracy respectively with 5,000 tagged sample tweets. With a much larger and more varied sample data, we should be able to make our algorithm more accurate for any context of the tweet. With a large enough tagged training data, will be able to track other kinds of real-life trends,

from preference of trends to political and ideological affiliation.

Works Cited

Paul, Michael. “Twitter Flu Annotations”. University of Colorado Boulder. June, 2013.

http://cmci.colorado.edu/~mpaul/downloads/flu_data.php

Python-twitter - Read the Docs - python-twitter 3.4.1 documentation.
<https://python-twitter.readthedocs.io/en/latest/index.html> Accessed March 15th, 2018.

Scikit-learn: machine learning in Python - scikit-learn 0.19.1 documentation.
<http://scikit-learn.org/stable/index.html> Accessed March 15th, 2018.

K. Zhang, R. Arablouei, and R. Jurdak. “Predicting Prevalence of Influenza-Like Illness in Australia From Geo-Tagged Tweets,” In proceedings of the 1st International Workshop on Social Computing (IWSC), as part of the 26th International World Wide Web (WWW) Conference, Perth, Australia, April, 2017.