Tweets Categorization and Sentiment Analysis

Abhisek Mohanty

Abstract*—In the past few years, there has been a huge growth in the use of microblogging platforms such as Twitter. Twitter and Facebook are not just portals for the public to put forth their feelings but also a platform which offers organizations a very effective way to get to know the response of the people to their brand, products and ventures. From e- commerce giants like Amazon to day to day traders and stock brokers can make use of twitter and online data to get an idea of which product is popular among the masses and which company stocks might possibly go up. In this project I propose to classify the content of the top tending topics on twitter and analyze their sentiment. Each tweet will be classified into a category, for e.g. Sports, Music, Technology etc. with the sentiment of the tweet as positive or negative. My aim is also to implement the above objectives using machine learning methods and find possible ways to leverage the findings in other projects and applications.

I. Introduction

In this age of information explosion, Micro blogging sites like Twitter are widely used and are a source of enormous data for a varied range of topics from finance to sports to entertainment as well as travel. As of 2015, about 500 million tweets are being generated every day ranging across a variety of topics. Twitter comes up with a list of trending topics every day and every hour. These trending topics are tailored by countries and cities and provide the users a lot of awareness on the hot topics of the day and moment. Now the important question here is, are these always useful? With fifty trending topics every hour of the day, the user will most likely not be able to figure out what the trending topics are about, solely based on the topic names. Hence, it is important to determine what these tweets are about and hence these topics need to be classified into broad categories like Music, Technology, Business, Travel, etc. The same can also be applied to the news articles from the links that the tweets contain. Moreover, with thousands of tweets being generated per second, it would be further beneficial and time saving if the tweets are classified into Interesting and non-interesting based on interests of the user.

Topic based classification is an important aspect for the user's point of view. In this age of information, with so much data, people generally tend to waste time trying to find the topics that they like. Hence a classification of tweets and articles based on the topic and person's interests is going to definitely find its place.

We then come to one of the most intrinsic properties of the tweets, which though very basic is one of the most important key performance indicator of a company and product - the sentiment. In technology, a positive tweet shows that a given product is well received by the consumers and vice versa, in Business, a positive tweet about a company indicates that the company is doing well and that there is a possibility that the stock prices might go up. Similarly, for Sports based articles, a positive sentiment of an article says that the team that the writer supports has probably won. In food products review sites and ecommerce websites like amazon, a positive sentiment of a product review directly translates to consumer satisfaction and can be used as a helping hand for other users when trying to decide on which item to buy.

Sentiment Analysis, which is basically an amalgamation of Machine Learning and Natural Language Processing, is widely used to understand people's opinion on a certain topic. Sentiment analysis is widely applied to reviews and social media interactions for a variety of applications, ranging from marketing to customer service to trading. These can be further used to generalize the interests of different people ranging from what smartphone people are more interested in to what vacation spot are people more interested in going to.

Generally speaking, sentiment analysis aims to determine the attitude of a speaker or a writer with respect to some topic or the overall contextual polarity of a document. Even though there is a possibility that the writer might be biased, we can eliminate that factor by analyzing various tweets on the same topic and then taking the mean, there by reducing the bias to a large extent.

II. Text Categorization

Text categorization refers to classification of text into predefined categories. Based on the attributes and the vocabulary, the documents are assigned a category they have the highest probability of belonging to. Given a document d_i, which is a tweet in this project, and classes C_j's, the text categorization classifier returns a c, and element of C_j to which

the document d has a higher probability of belonging to. There are various types of categorization problems, text classification being one of them. There is document classification, where an unknown is classified into image, text, video file or music file based on some intrinsic property of the file. Moving further, there is image classification where in the images are annotated with a text, which describes some aspect of the image, for ex. Sunset or Beach or Mountains. This has been implement in a lot of applications by a lot of companies, a common example being Google Photos image classification. Text categorization has various sub problems, some of which are email spam filtering and sentiment analysis. The concept classes vary for each of the sub problems. In a spam filtering problem, the text needs to be classified into spam or not spam. Likewise, in Sentiment analysis, the text needs to be classified into positive and negative sentiment.

This is a supervised learning problem where a classifier is trained on data that's annotated with a label representing the category the text belongs to. There are numerous machine learning techniques that can be used to categorize the text. Some of these are Naïve Bayes, K-Nearest Neighbors, Random Forests, Term Frequency-Inverse Document Frequency (tf-idf) and Support Vector Machines (SVM) which use the Bag of Words model where each term in the text is represented by its frequency.

As part of this project, the text categorization engine is implemented using Bag of Words model and tf-idf using Support Vector Machines and Naïve Bayes as the classifiers. Each tweet will be categorized into one of the fourteen categories, that are - sports, tech, travel, food, politics, automobiles, weather, music, entertainment, business, shopping, science, health or property.

III. Sentiment Analysis

Sentiment Analysis, also known as opinion mining, widely used in the computer science domain primarily refers to determining the polarity of the document, text or speech based data. The data is classified as positive, negative or neutral based on different attributes and the vocabulary of the data. In deeper sentiment analysis of data, the data is further classified based on the emotional states into happy, sad, angry etc. Sentiment Analysis can be seen as a sub problem of the Text Categorization problem where in the C_j classes of the text categorization problem are narrowed down to three to five classes, like positive, negative, neutral and so on.

There are many ways to do a sentiment analysis of data ranging from keyword based sentiment score to more complex classification using various machine learning techniques. Broadly the sentiment classification process can be divided into three categories – statistical techniques, keyword based methods and other hybrid approaches. The most basic is the keyword based method that banks on high intensity sentiment keywords like happy, sad, good, bad etc. and provide each keyword in the vocabulary with an appropriate sentiment score. These sentiment scores are then accumulated on tokenized text vocabulary to get an approximate sentiment of the text.

The other way of doing sentiment analysis is based on statistical methods where the text is classified based on machine learning classifiers. Most widely used machine learning techniques include latent semantic analysis, the bag of words model and support vector machines classifiers.

IV. Machine Learning Methods

Since the text categorization problem is inherently a supervised learning classification problem, there are various supervised machine learning methods that can be used to work it out. Some of the most commonly used learning algorithms are Naïve Bayes, k Nearest Neighbors, ANNs and Support Vector Machines. In this project, I have tried to implement the problems using Naïve Bayes and Support Vector Machines. The algorithms rely on the Bag of Words model where each word is represented by its frequency or inverse document frequency, disregarding the grammar and its order in the sentence.

1. Naïve Bayes:

Unlike Support Vector Machines which label the data to a particular class, Naïve Bayes is a probabilistic classifier which uses Bayes Theorem assuming that the features are independent. For a document d and a class C, the Naïve Bayes classifier arrives at the probability using the Bayes Theorem,

$$P(c \mid d) = \frac{P(c)P(d \mid c)}{P(d)},$$

where P(c | d) gives the probability of class c given the document d. Though a simple algorithm with assumptions of independence that is not always true, the Naïve Bayes classifier does a very good job of predicting the class of the document with a testing data accuracy of approximately 73%.

2. Support Vector Machines:

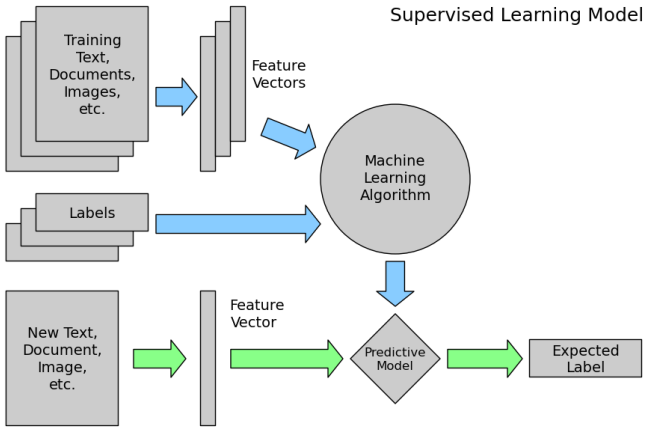
SVMs or large margin classifiers are one of the most widely used classifiers on classification problems. It is a supervised learning algorithm which defines a boundary with a hyperplane with the largest margin from the nearest points or the support vectors on each side. In addition to performing linear classifications, the support vector machines can perform non-linear classification using kernel and mapping the data points to a higher dimensional space. Unlike Naïve Bayes, which is a probabilistic classifier, SVMs are non-probabilistic, that is, given a test data to be classified using an SVM, it always assigns the data to one class or the other or multiple classes, if it's a multi class SVM.

This project uses SVM to accomplish the classification problem due to the following reasons:

Firstly, since we are doing a text classification in this project, the vectors generated as part of the document term matrix can be very high dimensional. This is not an issue with the twitter training data since each tweet is limited to 160 characters, but in the general case, the vectors can go as high as even hundreds of thousands. An SVM will be one of the best methods to handle this high dimensional feature space with ease.

Secondly, the document vectors are generally sparse and a support vector machine handles sparse vectors with ease compared to other classifiers. We can always use a kernel for extremely sparse vectors, producing a low rank matrix, which in turn improves the efficiency of the machine. We do not need to do this in this project since we never reach very sparse matrices with a character limit of 160 for each line.

We use scikit-learn SVM modules in this project. The flowchart of the procedure is depicted below.



The classifiers, support vector machine and Naïve Bayes in our case, are run on labeled training data, which is obtained from Twitter. The training data is first processed into feature vectors, represented by the tf-idf values. This is explained in the next section. The vectorized data is then fed into the classifier along the class labels for each row. Once the classifier is trained on the data, the model is run on real time tweets to check the expected label.

V. DATASETS

The datasets for both the parts of the project – the tweet categorization and the sentiment analysis is based on data from twitter. The twitter API is used to get the training data for the text categorization. Since the twitter API limits the number of tweet requests to 180 in a 15-minute window, the python program goes to a 15-minute sleep after every 180 requests and then retrieves the tweets again. This process thus takes a long time, approximately five to six hours to retrieve all the tweets and hence we store all the tweets into respective files that are described next.

The training data used in the text categorization module are retrieved using the twitter API. For each category, the tweets are retrieved from approximately five to seven twitter users that are associated with the selected category. For ex., users contributing to sports training data would be twitter users related to sports like @espn, @foxsports etc., @engadget, @TechCrunch for tech training data, @cnnpolitics and @nytpolitics for politics data and so on. The training data sources, i.e. the users for each category can be found in the categorization_training_data.py file of the project files. Once the data for a particular category is retrieved from twitter, the data is labelled with the respective category name and stored to disk as a json file with the name as <category_name>.json.

Each json file corresponds to a python dictionary with keys – category and tweets. In this project there are 14 <category_name>.json files in the folder train_data which is used for training the text categorization SVM classifier. Following is a head () of the automobiles.json file.


```
{
  "category": "automobiles",
  "tweets": [
    "We ve spotted the next Vauxhall Insignia testing again",
    "Has the time come for Mini to launch the Rocketman
Find out why it s more likely than ever",
    "See what s heading to the Beijing motor show from the
local manufacturers"
  ]
}
```


The sentiment analysis module uses the Sentiment_Analysis_Dataset.csv dataset that's available on the web. This dataset has the following attributes/columns:

1. the polarity of the tweet (0 = negative, 2 = neutral, 4 = positive)
2. the id of the tweet
3. the date of the tweet
4. the query. If there is no query, then this value is NO_QUERY.
5. the user that tweeted
6. the text of the tweet

The sentiment analysis classifier uses the columns 1 and 6 for classification – the polarity and the text of the tweet. One of the most important aspects to look at is the length of the tweet. Unlike movie reviews on imdb or user product reviews on amazon, which have no limit on the length of the reviews, the length of the tweets are limited at 160 characters, the average length of being 68 characters. So, it may not be always possible that every tweet contains an emotion or a polarity factor to be able to assign a specific sentiment. Therefore, a large lot of tweets may not be assigned to a positive and negative and hence it become crucial to have a neutral class in this scenario.

Most of the tweets have a lot of words from different language domains, many of which are short hand representations of the actual words. Most of tweets are also just used as a placeholder for a hyperlink which redirects to the main article. These sort of anomalies, particularly the special characters, hyperlinks, twitter names, numbers and other words in user retweets reduce the efficiency of the classifier and hence need to removed.

VI. Procedure

To prepare the data for the classifier, the data from all the 14 files is read one by one and the content is added to the main dataset along with the label describing the category it belongs to. This is done in the categorization_process_traindata.py file. A head(6) of the data_file is added below, where, in each item, the first entity, the number is the class and the second entity is the tweet.

['10', u'the Sample Sale April 19 20 Tue 8 6pm Wed 8 5pm at 358 5th Avenue']
['10', u'Doris Roberts Everybody Loves Raymond star dies at 90']
['10', u'Join Fashionrecycle at drop off gently used clothes and get 30 off your ENTIRE purchase through April 24']
['10', u'Rejoice UES New Yorkers are now open on 1282 3rd Ave 73rd amp 74th st ShoppingUES']
['10', u'The London Sample Sale April 19 24 at 260 Fifth Ave']
['10', u'I still can t decide who to vote for']

Given the training data is in English alphabet, it requires to be converted to a readable vectorized format before feeding it to the classifier. The first step is to tokenize the tweets into words to form a vocabulary and then remove the stop words from the vocabulary. Stop words are the most common words in the language, for e.g. the, is, at, which are the stop words in the English language and do not make much sense when training a classifier. This project uses the NLTK stop words corpus.

We then find the term frequency and inverse document frequency (tf-idf) of each of the terms. Tf-idf is a weighting scheme that is used to compute the weight of the term in a corpus based on the frequency of the term. Tf-idf basically shows how important is a word to a document.

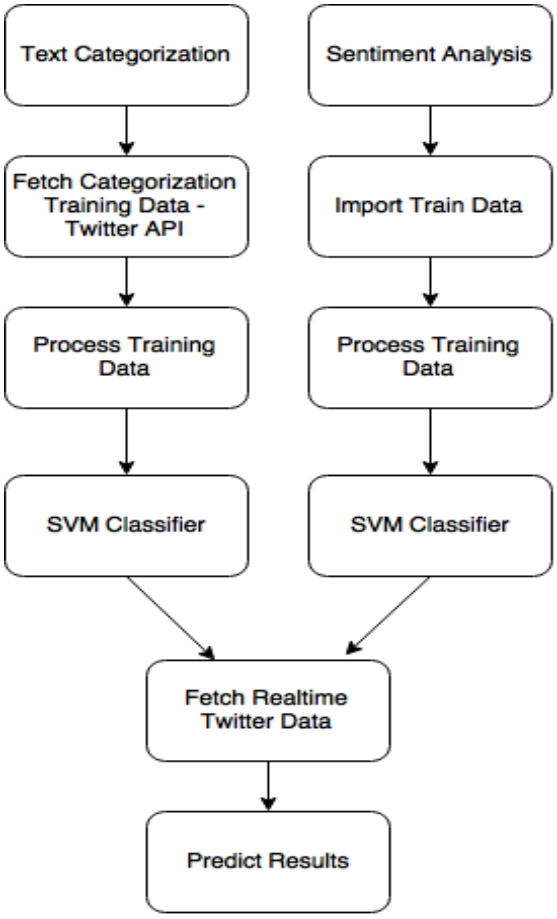
The next step is to convert the input data to a document term matrix. A document-term matrix is a matrix that describes the frequency of the terms that occur in a set of documents. A document-term matrix based on the above twitter data will consist of one row for each tweet. The columns will consist of:

a. the *index* of the word followed by the *inverse document frequency (idf)* of the word in the overall vocabulary file.
b. The frequency and the index are separated by a colon (:)

A row from the document-term matrix is shown below.

0 144:0.861775697771 294:0.759965616452
4208:0.861775697771 2750:0.861775697771
695:0.816724019093 921:0.861775697771
3162:0.861775697771 461:0.861775697771
2898:0.861775697771

The first 0 denotes the class of the tweets, i.e. the category it belongs to. The rest of the row makes up the document-term matrix. The values before the colon, 144, 294, 4208 etc. represent the indices of the words in the vocabulary list. The values following the colon are the inverse document frequency values. The above document-term matrix is computed in the tf_idf.py file. This vectorized data is then used in the training step for the support vector machine. A flowchart of the procedure is attached below.



The sentiment training data is also converted into a similar document-term matrix with a few changes. While using only unigrams in the categorization classifier is alright, it might not work in the sentiment analysis classifier. Let's suppose the tweet – u'that's was a bad movie', which clearly suggests it's

a negative sentiment tweet. In a naive unigram approach, the classifier learns that any tweet with above words is negative. Another tweet comes up in the test set – u'that was not a bad movie' – which is clearly a positive tweet, but a unigram based classifier classifies it negative, since this tweet consists of 2 highly negative polarized words. Hence, the need for using ngrams arises. Since we remove the stop words, a bigram model will consider both words 'not bad' and learn that the 'not bad' related to a positive sentiment and so on. Hence the prediction on the test sets increase.

VII. Evaluation

The two support vector machine classifiers trained in the previous steps, the text categorization classifier and the sentiment analysis classifier give an accuracy of 0.79 and 0.75 on the test set after a test train split of 0.9. This is using a Radial Basis Function(rbf) kernel. A linear kernel gives a similar accuracy while a Gaussian kernel decreases the accuracy by a large margin to about 50 percent. The same split on the dataset using a Naïve Bayes classifier gives an accuracy of 0.73 and 0.74 for the text categorization and sentiment analysis sub problems. The Naïve Bayes classifier though very simple and basic in comparison with the support vector machine, works pretty well for this problem.

The next observation was the average sentiment by country after the model was run on real time data fetched from twitter. While the average sentiment is very subjective, i.e. its influenced by a lot of factors ranging from the date and time the tweets were retrieved on, to the trending news occurring in the country at that time, the algorithm came up with the following top and bottom five countries. Top countries with the highest sentiment are:

Ireland 0.886435
Poland 0.890062
Germany 0.903444
Jordan 0.909753
Thailand 0.913845
Japan 0.924222
Kuwait 0.950249

While the countries with the lowest sentiment are:

Venezuela 0.716686
Denmark 0.720755
Colombia 0.721008
Indonesia 0.723893
Ecuador 0.724926
Belarus 0.725950

Also, an observation that might not be very relevant is related to the data retrieved from twitter. Since, twitter data comes with a lot of unnecessary characters and jargon, one very important step was to clean the data properly. With two trials, one without cleaning the twitter data (the data retained links, names and other characters), the accuracy dropped to 46%, which suggests that with more complex algorithms to clean the data we could have even higher accuracy with the model.

Two basic but computationally heavy improvement that could be done would be to ignore and discard words that are not in the English dictionary, and correcting the words that might be wrongly typed. The Levenshtein distance against a corpus of English words from the NLTK package can be used to correct the misspelt word and replace it with the nearest English word.

VIII. Related and Future Work

.

This project is a continuation of another project where I have worked on comparing the performance of difference machine learning algorithms like Naïve Bayes, SVMs, Random Forests, Deep Learning and k Nearest Neighbors (kNN) on datasets from UCI repository. I found that SVM is one of the most reliable methods for classification of data and hence planned to take up a practical project of classification based on Support Vector Machines. I also wanted to compare Naïve Bayes with SVM when trained on real world data, since Naïve Bayes always came up with results at par with other classifiers when run on smaller datasets.

I would like to extend this work further to another project related to news text summarization that I have been working on. I would be using the above algorithms to classify news articles into a wide variety of categories, which would include sub categories as well, e.g. soccer, football, tennis in sports and movies, television series, comedy in entertainment. It would also classify the news articles into interesting or non-interesting based on the news articles the user visits most. The only difference from this project would be the datasets which is comparatively easier to handle compared to twitter data and is based off a few web scrappers.

IX. Summary and Conclusion

As part of this project, I have worked on implementing two text classification engines using support vector machines, based on twitter data which can be used to classify the tweets into categories and also label the tweets as positive, negative or neutral based on their sentiment, got a hands on with the twitter API, which is very restricting, and hence had to find a way to program the restriction so as to retrieve a million tweets without any interruption. Had a few new observations regarding Naïve Bayes, which on paper seems simple but has a pretty good accuracy.

Working on this project has primarily helped me get an in-depth understanding of how SVM works and why it is better than other state of the art algorithms for text classification, which is highlighted in the report. This project introduced me to the difference between basic data classification (based on

numeric data) and text based classification. Text based classification needs a lot more emphasis on cleaning the data, especially with twitter data which has an abundance of unnecessary jargon and reduces the efficiency of the classifier.

REFERENCES

[1] Thorsten Joachims ,"Text Categorization with Support Vector Machines, Learning with Many Features"

[2] Yutaka Sasaki, University of Manchester, "Automatic Text Classification"

[3] Apoorv Agarwal Boyi Xie Ilia Vovsha Owen Rambow Rebecca Passonneau, "Sentiment Analysis of Twitter Data"

[4] Efthymios Kouloumpis, Theresa Wilson, Johanna Moore, "Twitter Sentiment Analysis: The Good the Bad and the OMG!"

[5] Bo Pang and Lillian Lee, Shivakumar Vaithyanathan, "Thumbs up? Sentiment Classification using Machine Learning Techniques"

[6] Kathy Lee, Diana Palsetia, Ramanathan Narayanan, Md. Mostofa Ali Patwary, Ankit Agrawal, and Alok Choudhary, "Twitter Trending Topic Classification "

[7] Hassan Saif, Yulan He and Harith Alani, "Semantic Sentiment Analysis of Twitter"

[8] Theodoridis, Sergios; and Koutroumbas, Konstantinos; "Pattern Recognition", 4th Edition, Academic Press, 2009

[9] Zach Chase, Nicolas Genain, "Learning Multi-Label Topic Classification of News Articles "

[10] István Pilászy, "Text Categorization and Support Vector Machines"

[11] Fabrizio Sebastiani, "Machine Learning in Automated Text Categorization"

[12] Susan Dumais, Decision Theory and Adaptive Systems Group, Microsoft Research, "Using SVMs for Text Categorization"

[13] Simon Tong, Daphne Koller, "Support Vector Machine Active Learning with Applications to Text Classification"

[14] Alec Go, Richa Bhayani, Lei Huang, "Twitter Sentiment Classification using Distant Supervision"