Twitter Hashtag Prediction

Abstract:
Social media has shown immense growth in form of micro blogging site like twitter, Flicker and Delicious which uses a unique tag system for navigation and hypertext browsing. Twitter network is currently overwhelmed by massive amount of tweets generated by its users. For each post or tweet user can have hashtag attached to it. Here in this paper we have design an algorithm to predict hashtag attached to a new tweet. Every tweet has various different noises which comes out to be challenge for us. We have implemented various preprocessing techniques to remove noise in the data and implemented an effective method of hashtag prediction based on popular classification algorithm like Naive Bayes and K nearest neighbor.

Introduction:

In recent years we have many new micro blogging sites are coming up which got very popular in very short span of time. Twitter is one them, currently it has 302 million users who posts around 500 Million tweets per day. Twitter has grown beyond a more than micro blogging site to become social messaging platform. In 2009 twitter introduced metadata tag called hashtag which allows users to connect and engage with other users based upon common theme and interests.

A hashtag is a word prefixed by # and we can have more than in one tweet. Currently hashtag is likely the most popular means of categorizing content on social media. It makes your own content discoverable and allows you to find relevant content from other people and businesses. Twitter allows users to hashtag their tweets and classify them and in this algorithm we have used machine learning and data mining techniques to classify the tweets according to the hashtag and predict a hashtag for new incoming tweet. We faced various challenges like tweet is too short and it contains miss spellings, abbreviations, noise and incorrect grammar.

With traditional document classification techniques such as TFIDF it's difficult to calculate importance of word in tweet because in some cases there could be a term which has occurred only once, so term frequency for these words is one and these rarely repeated words in tweet comes out to be most important one which makes this technique ineffective. To tackle these challenges we are performing various preprocessing techniques and did prediction based upon popular classification algorithm like Naive Bayes and K nearest neighbor.

Related Work:

The data set in social networking sites like Twitter is updated very frequently due to the activity of millions of Twitter users online. Due to this dynamic change in the data set, it is difficult to correctly predict the hashtags for tweets. A distance function to classify and predict hash tags is discussed in [1]. This approach uses machine learning technology where collected tweets are mapped to a high dimensional space and a latent network is constructed to predict the similarity of these tags.

A personalized hashtag recommendation method that considers both user preferences and tweet content is proposed in [2] where for a given user and a given tweet, we find the top most similar users and top most similar tweets. Hashtags are predicted from most similar tweets and users by assigning ranking scores and selecting the highest of the scores assigned.

In [3], three classifiers are proposed for predicting hashtags in twitter. A Naïve Bayesian Classifier is formulated using words in the tweets as features and class as the hashtag. It calculates the probability of a hashtag given a set of words in the tweet. This is done for all known hashtags and the one with highest probability is suggested as the recommended hashtag. K-nearest neighbor classifier is used which uses the most similar k-neighbors using Term Corpus Relevance technique and predicts the hashtag for a given tweet. Also a combined classifier using both naïve bayes and k-nearest neighbor classifier is proposed to improve accuracy. We are using these approaches in our hash tag prediction algorithm.

Proposed Method:

In this we proposed a method to predict hash tag for incoming tweet and process is divided into several parts data preprocessing, machine learning classification.

1) Data Preprocessing:

a) Eliminating words: - Twitter Data may contain some noisy data. It could be in the form of abbreviations, spelling mistakes, incorrect grammar and hyperlinks. So Preprocessing consists of filter, aggregate, cleansing the dataset to make it efficient. We have removed URLs, special HTML entities, digits, punctuations and various other non-alphanumeric words. To maximize the likelihood of our classifier, we removed words that provide very little to the meaning of the Tweet. We also removed the words which are less than three characters.

B) TF-IDF (term **frequency**–**inverse document frequency) - It** is a numerical statistic that is intended to reflect how important a word is to a **document** in a collection. We have implemented TF-IDF to find the similar tweets. For a given tweet, we take all the words in it and calculate the value

$$\log \frac{|Tweets|}{|word \in tweet| \,|}$$

From these we take the three words which has the maximum value to calculate the similar tweets.

C) Word Weight computes additional weighting for a given word. We apply a weight of 3 to all words containing '@', because tweets directed to the same user have a high likelihood of being similar and containing the same hash tag. The bigger the word, the more importance it has. Hence we use the length of the word in addition to the word weight.

2) Classification: Our prediction basically works with classification, so we have evaluated different classification techniques and finally decided to use naive Bayes technique and k nearest neighbor because it is easier to implement and also we felt that this would give us the best results for this particular problem based on our findings. Both algorithms are lazy evaluating algorithms. So in this the classification model can be modified without needing to rebuild the entire model.

a) Naive Bayes: We used naive Bayes classification which basically uses Bayes theorem to determine the conditional probability of a class given an item's feature. So in our system the class we have defined is hashtag and the features are the words in tweet after screening and cleaning. So by using Bayes theorem we calculate the probability of hashtag given set of words in tweet. We do this for all the hashtags and these are used in predicting hashtags for incoming tweet.

Theorem 1 (Bayes' Theorem). Multiple Features

$$P(h|w1,w2,\dots,wn) = P(w1,w2,\dots,wn|h) * \frac{P(h)}{P(w1,w2,\dots,wn)} = = P(h|w1) * P(h|w2) * \dots * P(h|wn)$$

However, with this approach we could get into some cases where, $P(h|wi)$ will be zero when the tweet doesn't have that particular word. So the total probability $P(h|w1,w2,\dots,wn)$ will become zero even if other words have high conditional probability. So instead, by summing the conditional probabilities, we preserve the high weighting contributed by the high probability words. For every tweet, we find the list of similar tweets using TF-IDF (Term Frequency - Inverse Document Frequency). Then from the list of tweets, we get all the hashtags. For every hashtag in the list of tweets, we find the probability score. For the probability score we also use weights for each particular word .Finally we give the top two hashtags based on the score.

Algoriithm1
[BAYES_CLASSIFY(Words,Tweet,n)]

```
T_s = SAME_TWEETS(Words,Tweet)
Hashtags = T_s..hashtags
Class = []
for itr in Hashtags do
    score = 0
    for w in Words do
        score += (P(w|h) * P(h)/P(w) * WORD_WEIGHT(w))
    end for
    class.add(h,score)
end for
return get_highest_score(class,n)
```

b) K-Nearest Neighbor Classifier

We have implemented the K-nearest neighbor classifier to predict the hash tag for a particular tweet. In this approach, we will find most similar tweets from the training data set for the particular tweet. Term-Corpus Relevance (TCoR) is used to find the similarity measure. TCoR is a weighting measure to measure how strong of a class predictor the word is across the data set. It is calculated using for a word in the tweet using the below formula.

TCoR(w) = ((1/fl(w)) + (1/c(w)))/2

Here fl(w) is the average number of words in tweets containing the word and c(w) is the number of hashtags the word co-occurs with. The intuition for TCoR is that the fewer number of words in a tweet, the more important an individual word is to the overall meaning of that Tweet and words occurring with a small number of distinct hashtags are more informative when predicting a hashtag. In k-nearest neighbor classifier, we are getting the list of similar tweets for the test tweet using TF-IDF (Term Frequency - Inverse Document Frequency). In the list of similar tweets obtained, we are calculating a score for each tweet in the list using TCoR and word weight calculated for every word in the tweet. We get the top k tweets having the highest score from the list. After finding the list of k tweets we are interested in, we find the counts of all the hashtags from these k-tweets. The hash-tag which occurs the maximum will be the hash-tag predicted for our test tweet.

ALGORITHM:

W – List of words in the test tweet
T – List of tweets in training set
k – Number of nearest neighbors to consider
n – Number of hashtags to predict

```
HASHTAG_PREDICTION USING K-NEAREST_NEIGHBOR(W,T,k,n)
        ST = Similar_tweets(W,T)
        Neighbour_tweets = []
        For each tweet t found in ST
                Score of tweet = TCoR(W) * Word_Weight(W)
                Neighbour_tweets.add(t,score)
        End for
        Nearest_Neighbours = GetHighestScores(neighbors,k)
        Predicted_hastags = []
        For t in Nearest_Neighbors
                For h in t.hashtags
                        Predicted_hashtags.add(h,1) //Add h with count of 1 or incremented count
                 End for
        End for
        Return getHighestScores(Predicted_hashtags,n)
```

3. Combining Naïve Bayes and K-Nearest Neighbor

To take advantage of the results of each classifier, we combine both naïve Bayes and k-nearest neighbor classifier. We combine both these classifiers so that the most likely hashtags are moved to the top of the list of recommendations. Each hashtag recommended by naïve Bayes receives a weight of 0.4 and each hashtag recommended by k-nearest neighbor receives a weight of 0.6. We sum up the weights and return the hashtags with the highest score. This way, a hash-tag which is highly recommended by both the classifiers is given more importance than a hash tag highly recommended by only one of them.

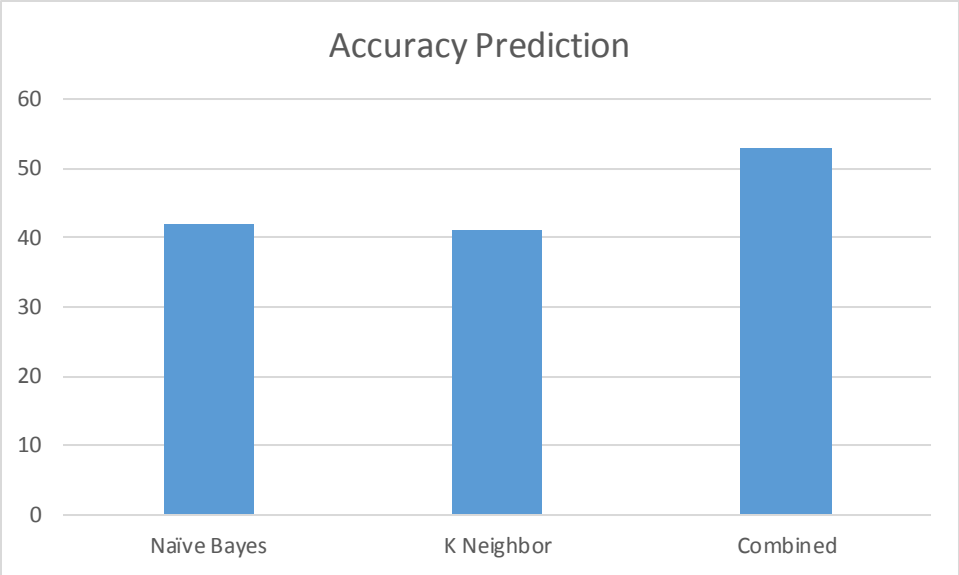
4 Results

We calculate the accuracy of each classifier by finding the fraction of the test tweets for which the hash tag is predicted properly out of the total test tweets considered. From the data provided, we have considered 550000 tweets (approximately) for the training data set and 100000 tweets for the test data set. For each tweet in the test data set, we ran all the three classifiers and have found the accuracy for each of them. Each of the classifier uses the training set to predict the hash-tag for all the test tweets present in the test data set. We have listed the accuracy of each classifier in the table below.

Accuracy is calculated by finding the number of test tweets containing one of the two hashtags predicted by our classifier and dividing by the total number of test tweets

Accuracy:

Classifier	Accuracy
Naïve Bayes	42%
K-Nearest Neighbor	41%
Combined Classifier	53%



5. Conclusion:

Thus we propose three different approaches for hashtag prediction. We found that K-nearest neighbor classifier prediction is similar to naïve Bayes classifier. We have also tried to combine both these approaches to get a better prediction rate. It can be seen that the combined approach gives us much better accuracy than the K-Neighbor or Naive Bayes. We have faced some challenges like finding the most similar tweets to a given tweet and also in preprocessing stages. We feel that this approach will work much better if we could find a better way to find similar tweets.

6. References:

1) Twitter Hash Tag Prediction Algorithm, Tianxi Li and Yu Wu, Department of Computer Science, Stanford University, (tianxili, ywu2}@stanford.edu, Yu Zhang,Department of Computer Science, Trinity University,yzhang@trinity.edu
2) On Recommending Hashtags in Twitter Networks, Su Mon Kywe, Tuan-Anh Hoang, Ee-Peng Lim and Feida Zhu Singapore Management University, Singapore {monkywe.su.2011,tahoang.2011,eplim,fdzhu}@smu.edu.sg
3) Twitter Hash Tag Recommendation, Roman Dovgopol and Matt Nohelty, University of Minnesota
4) Using TF-IDF to Determine Word Relevance in Document Queries, Juan Ramos, juramos@eden.rutgers.edu, Department of Computer Science, Rutgers University