Sensing Topics in Tweets

Amar Budhiraja
Centre for Data Engineering
IIIT-Hyderabad
Hyderabad, India - 500032
Email: amar.budhiraja@research.iiit.ac.in

Manas Tewari
Centre for Exact Humanities
IIIT-Hyderabad
Hyderabad, India - 500032
Email: manas.tewari@research.iiit.ac.in

Vikram Ahuja
Centre for Exact Humanities
IIIT-Hyderabad
Hyderabad, India - 500032
Email: vikrm.ahuja@research.iiit.ac.in

***Abstract*—Identifying topics in Tweet in well established problem because of their short length, noisy data due human errors and uneven lingo used across social media. In this report, we explore different approaches to identify tweets of similar topics. We use three Tweets datasets pertaining to three different domains. Through our research, we found that NAME OF APPROACH performs the best. We also augment the used approaches by boosting proper nouns and then, recalculating the experiment results on the same three datasets.**

I. Introduction

With the advent of Social Media, enormous amount of information exists which has been and can be mined for interesting results. Apart from several applications, mining data from social Networks like Twitter has results in helping with natural disasters, city management and political opinion mining for prediction of Elections. Identifying topics corpora has been an interesting problem for NLP researchers for long. For most interesting applications, it becomes important to correctly detect the topic of a Tweet. However, Tweets being only 140 characters long present a rather intriguing challenge for researchers. In this report, we aim to perform topic detection across three different topics of Tweets extracted from three different events. We perform topic clustering among the data using LDA (Latent Dirichlet Allocation), Naive Bayes, Clustering using K-Means, K-Nearest Neighbors and Boosted N-grams. Section II elaborates on each of the approaches. Section III discuss our setups and experiments including the dataset, libraries and results. Section IV provides a discussion on the approaches their relative performance. In Section V, we provide conclusions and how we plan on taking this work

II. Approaches Employed

In this section, we discuss the approaches that we have employed to perform topic detection in Tweets. We have employed Naive Baye's, LDA, K-Means Clustering, k-Nearest Neighbor and boosted N-grams.

A. *LDA*

Topic extraction in textual corpora can be addressed through probabilistic topic models. In general, a topic model is a Bayesian model that associates with each document a probability distribution over topics, which are in turn distributions over words. LDA is the best known and most widely used topic model. According to LDA, every document is considered as a bag of terms, which are the only observed variables in the model. The topic distribution per document and the term distribution per topic are instead hidden and have to be estimated through Bayesian inference. In simpler words, LDA provides a transitive relationship between words such that each document has a set of words and each word belong to multiple topics which transitively indicates that each document is a collection of topics.

B. *Naive Bayes Classifier*

Naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features. For Tweet classification, we implement Naive Bayes as a simple Bag of Words approach. For a given Tweet t, the probability that it belongs to class c (from Bayes Theorem),

$$P(c|t) = \frac{P(t|c) \times P(c)}{P(t)} \tag{1}$$

Maximum a posteriori probability i.e. most likely class is given by,

$$c_{MAP} = argmax P(c|t) \tag{2}$$

$$c_{MAP} = argmax \frac{P(t|c) \times P(c)}{P(t)} \tag{3}$$

Denominator can be removed as it acts like a scaling factor,

$$c_{MAP} = argmax P(t|c) \times P(c) \tag{4}$$

Since, each Tweet is modelled as a bag of words,

$$c_{MAP} = argmax P(x_1, x_2 ... x_n|c) \times P(c) \tag{5}$$

Occurrence of each word and its order in the Tweet is irrelevant so,

$$c_{MAP} = argmax P(c) \times P(x_1|c) \times P(x_2|c) \times ... P(x_n|c) \tag{6}$$

$$c_{MAP} = argmax P(c) \times \Pi P(x_i|c) \tag{7}$$

Hence, for each Tweet the c_{MAP} is the topic of the Tweet.

C. *Clustering using K-Means*

Clustering algorithms aim at finding the natural grouping of a given set of elements with a notion of distance between points so that members of a cluster are close/similar to each other and members of different clusters are dissimilar. Clustering is generally an unsupervised learning task. For two vectors x_1 and x_2, different similarity models pertaining textual vectors have been developed including :

$$CosineSimilarity = \frac{x_1.x_2}{|x_1| \times |x_2|} \tag{8}$$

$$JaccardSimilarity = \frac{x_1 \cup x_2}{x_1 \cap x_2} \tag{9}$$

For our study, we have considered cosine similarity in the K-means method. Simply speaking, K-means an algorithm to classify or to group your objects based on attributes/features into K number of group. K is positive integer number. The grouping is done by minimizing the sum of squares of distances between data and the corresponding cluster centroid. Thus, the purpose of K-mean clustering is to classify the data.

K-means clustering algorithm

1) Start with initial guesses for cluster centers (centroids)
2) For each data point, find closest cluster center (partitioning step)
3) Replace each centroid by average of data points in its partition
4) Iterate 1+2 until convergence

D. *K-Nearest Neighbor*

K-Nearest Neighbors is an unsupervised approach in pattern recognition. It is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition.

For each iteration of KNN algorithm, distance of input point is computed with every other data point in the set and the K closest neighbors are noticed. Usually, a majority vote is taken into consideration to assign a class to the incoming point and hence, an odd value of K is preferred. Given query instance x_q, KNN works as follows,

1) First locate nearest training example x_n, then estimate $\hat{f}(x_q)f(x_n)$
2) Given x_q, take vote among its k nearest nbrs (if discrete-valued target function)
3) Take mean of f values of k nearest neighbors (if real-valued)

$$\hat{f}(x_q)\frac{\sum_{i=1}^{k} f(x_i)}{k}$$

III. Experiments

In this section, we first discuss the dataset used followed by our implementation methodology for each of methods discussed above and finally conclude with results.

A. *Dataset*

We have used three Tweet datasets - FA Cup Final, , Super Tuesday for US Elections and US Elections. These three datasets were extracted and worked upon in a event detection setting. We randomly selected tweets from each of the three datasets such that the number of tweet for FA Cup were almost 2000, for US Elections almost 2000 and for Super Tuesdays almost 2000.

This dataset is important because it has contrast in terms of FA Cup being a sports dataset and the other two datasets being politics oriented. The dataset also provide closeness in topics as both Super Tuesday for US Elections and US Elections and this helps us in establishing how the used algorithm will perform when the topics are close to each other.

B. *Implementation Methodology*

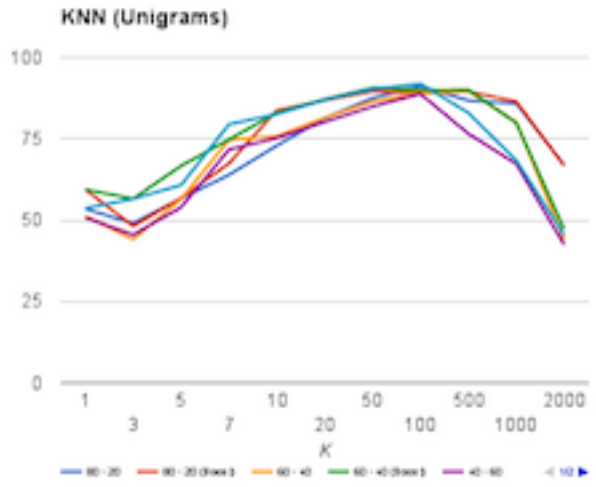
For LDA, we used a prebuilt Python Library. We have implemented Naive Bayes algorithm according to the assumptions stated in the earlier sections that each document is actually a bag of words and each words occurs in a class. We didn't perform any smoothing operation over Naive Bayes. If a word is not seen below, we do not consider it to be a part of the tweet.

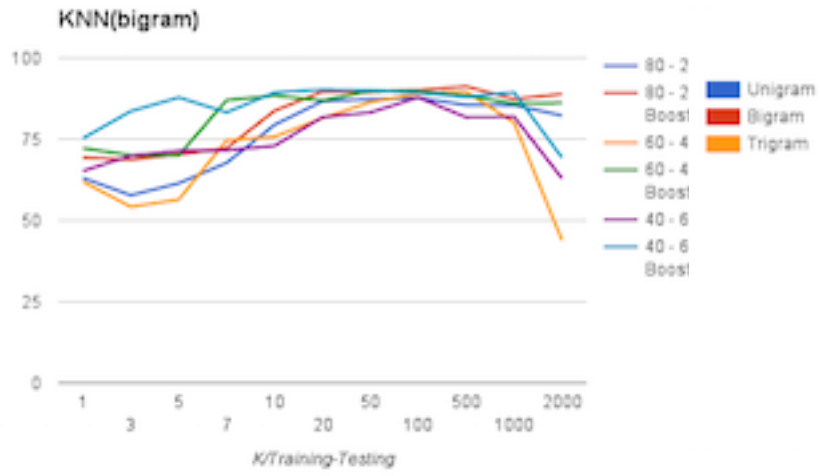
For K-means, we implement the method using cosine similarity using sparse matrix in Python. The number of clusters K was taken to be 3 i.e. the number of topics.

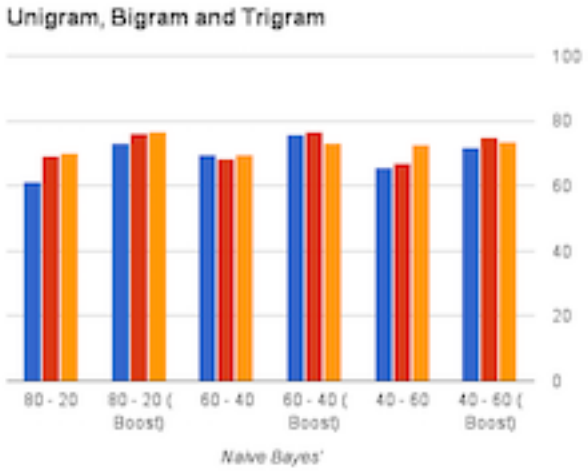
For KNN, we have used TF*IDF, and then compared the respective scores between training and testing data in order to classify the tweets.

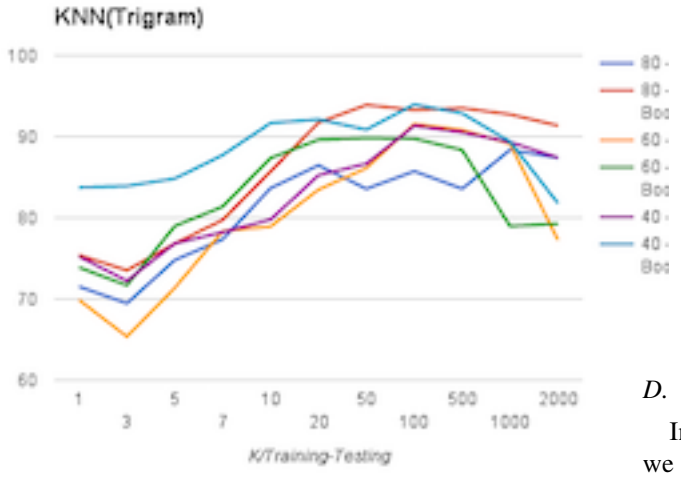
The boosting techniques that we have used are removal of stop words, also known as function words, duplication of hashtags and proper nouns. The boosting techniques were chosen keeping in mind the need to give priority to the tweet semantic.

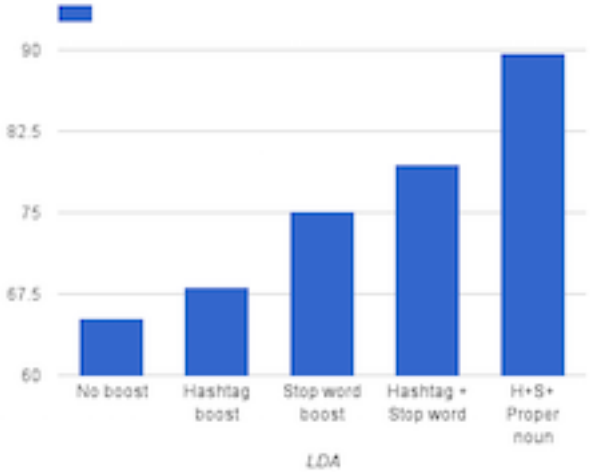
C. *Results*

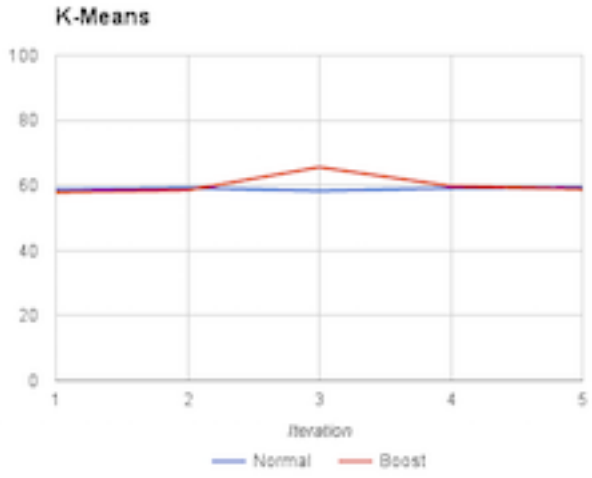












D. Conclusions

In this section, we discuss the results. Through this study, we noticed that semantics of tweets helped in enhancing the accuracy. It was made evident when Naive Bayes based on bigrams performed better then the Naive Bayes based on unigrams and the Naive Bayes based on trigrams performed better than the one based on bigrams. For all of the cases that we discussed, we also boosted the proper nouns and each of the classifiers we explored, the classifier with boosted proper nouns performed better in every case.

The worst performing classifier was K-means clustering, which was expected considering the fact that it is an unsupervised technique. However, LDA is also an unsupervised technique, but its performance increases drastically after boosting, due to the nature of boost techniques.

KNN's accuracy increases till a particular value of K and then decreases.

ACKNOWLEDGMENT

The authors would like to thank their Teaching Assistants and Dr Avinash Sharma for their invaluable guidance during

the commencement of the project.

REFERENCES

[1] Conover, Michael D., et al. "Predicting the political alignment of twitter users." Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on. IEEE, 2011.

[2] Phuvipadawat, Swit, and Tsuyoshi Murata. "Breaking news detection and tracking in Twitter." Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on. Vol. 3. IEEE, 2010.