Auto-Tag

Varun Ranganathan
Computer Science and Engineering
PES University
Bangalore, Karnataka- 560085
Email: varunranga1997@hotmail.com

Tanya Mehrotra
Computer Science and Engineering
PES University
Bangalore, Karnataka- 560085
Email: tanumehrotra01@gmail.com

***Keywords*—Natural Language Processsing, Deep Neural Networks, Naive Bayes Classification, NLTK, Hashtags, Tweets.**

I. Abstract

Hashtags or #hashtags are one of the new trending methods to organize data on the web. We are generating hashtags for a given tweet by the use of supervised machine learning algorithms. The two models used by us, to implement the same are Naive Bayes and Artificial Neural Networks. Data is preprocessed using Natural Language Processing, and then it is fed to the models. Our training data consists of NLTK dataset for tweets and the testing data comprises of random tweets using the Twitter streaming API, Tweepy.

II. Introduction

Social media outlets like Twitter, Facebook and Instagram allow the users to label their content, which in turn makes them accessible to other users. Of the above mentioned microblogging services, Twitter has emerged out to provide a dominant service with estimates of 250 million registered users who post up to 500 million Tweets per day. With the increase in the popularity of Twitter, the need for an organizational strategy to manage its content also increased. This is done by the use of hashtags, which gives a conglomeration of numerous pictures and tweets about the topic to which it is related. There is no limit to the type of hashtag that a user can create, unless and untill its a single alphanumeric string of text. Therefore, there is a vast pool into which the information can be grouped and accessed. But it is seen that only 10% of the tweets are tagged with hashtags, which somehow destroys their purpose. Thus, if there is a system in which hashtags can be recommended to the users as they are tweeting, more data can be tagged and the usage of hashtags can be increased dramatically, thereby increasing the audience of the content that is put on the web. The main advantage of using hashtags is enhancing the Twitter search. If a user searches for a keyword in the form of a hashtag, all the tweets containing that hashtag will be given as search results, which would give the user a wide spectrum to pick up the most relevant of them all. People are thus, exposed to different versions of related data, that can assist in better awareness.

III. Literature Survey - Summary

A. Kywe, Su, Tuan-Anh Hoang, Ee-Peng Lim, and Feida Zhu. On recommending hashtags in twitter networks. Social Informatics (2012): 337-350

The dataset was collected on Singapore users over a period of thee months, comprising of a total of 44M tweets. Users were represented by their preference weights for each hashtag. The hashtag weightage is given by TF-IDF method. Tweets were also represented as a weighted vector of words using the same technique. The probable hashtag that can be recommended for a user u and tweet t was obtained by taking the union of hashtags from top-X similar users and top-Y similar tweets. This new list of suggested hashtags was ranked by frequency and the top ranked ones were recommended.
The results depend upon the value of X and Y.

B. Godin, Frderic, Viktor Slavkovikj, Wes- ley De Neve, Benjamin Schrauwen, and Rik Van de Walle. Using topic models for twit- ter hashtag recommendation. In Proceed- ings of the 22nd International Conference on World Wide Web, pp. 593-596. ACM, 2013

Using the Twitter streaming API, 18 million tweets were collected. The dataset was pre- processed by removing URLs, special HTML entities, digits etc. Slang words were converted into proper english words. An unsupervised language classifier was used because the lan- guage of the tweet may or may not be the same as the locale. For Hashtag Recommendation, a binary classi- fier, based on Naive Bayes technique was implemented, which discriminated between En- glish and non-English language tweets. Latent Dirichlet Allocation (LDA), which is a hidden topic model, was used for retrieving information from a document by finding out the general topics in it. Gibbs Sampling Algorithm was used to find the undelying topic of a new tweet. From this topic distribution, keywords were selected and later used as hashtags.
The language classification algorithm converges to a local maxima and therefore the training set depends upon the starting conditions. Thus, while training, multiple versions of the classifier were fed with different starting parameters. The language classifier gave an acccuracy of 97.4of the tweets, at least one suitable hashtag could be suggested out of five recommended hashtags

IV. Problem Statement

Generation of Hashtags for a Tweet

We seek to address the problem of making the data more available to the users of Twitter, by increasing the scope of the content provided to them. This should be achieved effortlessly and be as accurate as possible. The simplest way to bring about this modification is to use relatable hashtags for a tweet, such that it can be associated with as many topics as possible, without losing its sense. In this project we provide a platform for the users to get a list of suggested hashtags for the tweets entered by them. The implementation is done in real time, using Tweepy, the Twitter's API, to fetch real time tweets.

A. Data Source

NLTK tweets are taken as our dataset. The NLTK dataset contains tweets of General Elections 2015, which are only used for training purposes because its specific to a particular topic. Since we aim to generate hashtags in real time, Tweepy, the Twitter's API is used for testing purposes. The tweet is extracted in the form of a json object. It is then preprocessed and then given to our models. A tweet consists of many fields, like location, timestamp, user details, URLs mentioned in the tweet etc. However, all this is not required for our model. We are only concerned with the text of the tweet written by the user and no other details. Therefore, we take care that only the tweet, i.e., the text in the tweet is recorded in the dataset and nothing more, just to reduce the complexity.

B. Assumptions/Constraints

To get a good accuracy for our project, we need a really huge dataset for its operation. Therefore, all the computations are done on Google Cloud and cannot be replicated on laptops. For a demo purpose as to how the models work, they can be demonstrated individually, with the NLTK dataset. The Naive Bayes Classifier and Support Vector Machine, is able to run on the laptop, when the input is restricted to 500 tweets. The DNN model, however, cannot be trained on laptop, but testing is possible.

V. Problems Faced

While brainstorming about what all methodologies can be applied to this problem statement, we came up with the idea of a Recurrent Neural Network with LSTM, with the same dataset, and a bagOfWords containing all possible words. However, the model could not meet up to our expectations. Maybe, because there was no need for the neural network to remember the past data since the implementation is real time. Thus, we finally narrowed ourselves down to three approaches of solving the problem - Naive Bayes, Support Vector Machines and Deep Neural Network.
Another change that we made was to replace the bagOfWords with the GloVe Dictionary, the difference being that bagOfWords contains words in alphabetical order but in the GloVe Dictionary, the ordering of the words is dependent upon their meaning. Words that can be associated are grouped together. Thus, it gave a more systematic platform for our model to actually understand the data, rather than performing simple training and prediction.

VI. Proposed System

A. Preprocessing

Various techniques of Natural Language Processing have been used to pre-process the data, for all our models - Naive Bayes, Support Vector Machines and Deep Neural Network. Post using the word tokenizer and tweet tokenizer, a lot of changes have been done to the tweets in order to make it convenient for the models to decipher its content. Apostrophe words like can't, don't, etc have been completed into their full words like 'can not' and 'do not' respectively. The words are then lemmatized, aiming to remove inflectional endings only and to return the base or dictionary form of a word. Next, stopwords are removed because they are not at all relevant and should not be taken into consideration while generating hashtags. Any forms of URLs present in the tweet are also removed. Lastly, slang language, which mostly consists of abbreviated words, are replaced by the complete, meaningful words.
Another of the methods used in preprocessing, is correcting the spelling of the misspelled words. Spelling corrector, made by 'Peter Norvig' has been adopted for the same - an edit to a word is carried out by deletion, transposition (swapping), replacement or insertion. Words are restricted to a limited vocabulary defined for this method and the number of edits made to a misspelled word, to convert it into a meaningful word from the vocabulary is taken into account. Corrections that require two simple edits are also taken into consideration, generating a much bigger set of probabilities, but not necessary all words formed are known words. P(word) is estimated by its occurences in a text file containing a million words, big.txt. It is a concatenation of public domain book excerpts from Project Gutenberg and lists of most frequent words from Wiktionary and the British National Corpus.This model gives around 68-75% accuracy.

B. Artificial Neural Network Model

An Artificial Neural Network is being used for the process of training the model to learn about the various hashtags that can be predicted by the network based on the content inputted by the user. The ANN is built on TensorFlow and implemented by Keras. There are 23920 input X vectors, each containing 400000 numbers. The vectors are based on the sample data available on NLTK on Python, and the Glove dataset, which is being used as the bag of words. The output vectors Y are one hot encoded, where the position of the digit '1' refers to the word at that index in a bag of hashtags. The model is a 4 layered network, with 512, 1024, 1024 and 3600 neurons at the respective layers. Each layer is activated by the sigmoid function, with the last layer activated by a softmax function. The optimizer used is Adam, with the losses calculated using categorical crossentropy. The model is made to train for 120 epochs in batches of 64 samples processed at a time. The learning rate employed by the network is 0.01, with 0.9 momentum. The training process is done on a Google cloud system, with specifications of 72 CPU cores, 416GB RAM, and a 100GB hard disk.

C. Naive Bayes Model

The input to the Naive Bayes classifier is a pickle file containing a dictionary, wherein the keys are the hashtags and the values are words of the corresponding cleaned tweet (stored in the form of a list). Tweets that do not contain a hashtag in the training set are omitted.

Naive Bayes Theorem states : $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$

Where,

1)P(A) represents the prior probability, or initial belief of A

2)$P(A|B)$ represents the conditonal probability of A given B.

3)$\frac{P(B|A)}{P(B)}$ represents the support B provides for A

We are examining the following conditional probability :
$P(hashtag/tweet) = \frac{P(hashtag)*P(tweets/hashtags)}{P(tweets)}$

The algorithm makes a naive assumption that all features are independent, given the hashtag :

$$P(hashtag|tweets) = \frac{P(hashtag)*P(t1|hashtag)*...*P(tn|hashtag)}{P(tweets)}$$

Rather than computing P(tweet) explicitly, the algorithm just calculates the numerator for each label, and normalizes them so they sum to one :

$$P(hashtag|tweet) = \frac{P(hashtag)*P(t1|hashtag)*..P(tn|hashtag)}{SUM[h](P(h)*P(t1|h)*..P(tn|h)}$$

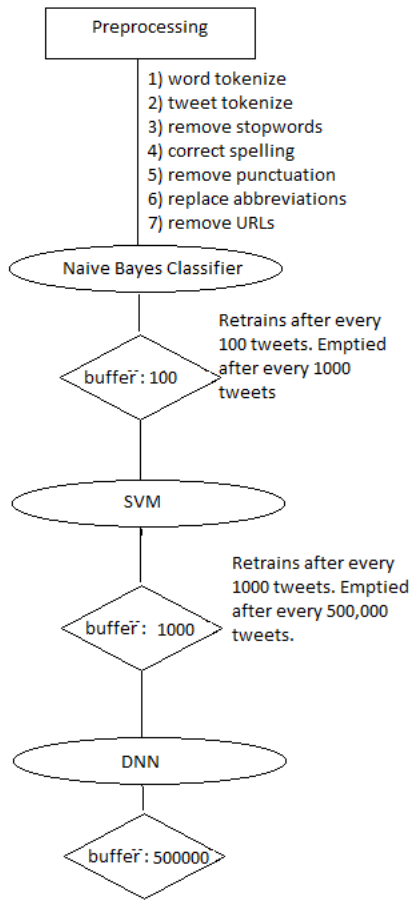
If the classifier encounters an input with a feature that has never been seen with any label, then rather than assigning a probability of 0 to all labels, it will ignore that feature. Hence, all tweets having no hashtags, are ignored.
Training is performed on the feature set and while testing, the most probable hashtag is returned.

D. Support Vector Machines

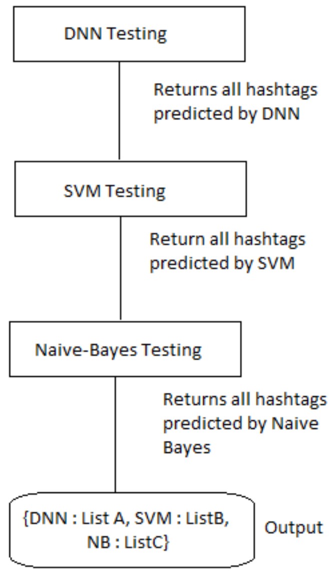
SVM is implemented as a wrapper around scikit-learn by constructing a scikit-learn estimator object. This is used to construct a SklearnClassifier. The text is automatically converted into vectors for SVM prediction. Its implementation is somewhat similar to that of Naive-Bayes. The input again, is taken in the form of a dictionary, where the keys are the hashtags and the values is a list containing the words of the corresponding cleaned tweet. While forming the dictionary from the tweets and their hashtags, if hashtag(s) is absent from a tweet, that tweet is ignored, for simplicity purposes.
Linear SVM is used as a function here, which takes longer to train that Naive Bayes Classifier.

E. Flowchart

1) Listener Process Flowchart



2) Advisor Process Flowchart


DNN Testing
Returns all hashtags predicted by DNN
SVM Testing
Return all hashtags predicted by SVM
Naive-Bayes Testing
Returns all hashtags predicted by Naive Bayes
{DNN : List A, SVM : ListB, NB : ListC}
Output

F. Combining all three models

Instead of choosing the mainstream method of running the models individually and comparing their accuracy, we came up with the idea of combining all three models and running them simultaneously. Each model has a buffer which indicates as to how many tweets it can train at a time. The order of execution of the models are Naive Bayes, Support Vector Machines and Deep Neural Network.
A buffer is set for all the three models :
Naive Bayes - 100 tweets
Support Vector Machine - 1,000 tweets
Deep Neural Network - 500,000 tweets
The order chosen for the execution of the three models is basically because of the time taken by each. Naive bayes takes the minimum time for trainingz so it is executed first, with the least buffer. Support Vector Machine, which takes more time than Naive Bayes to train, is kept second, having medium buffer. Lastly, Deep Neural Network, being the most accurate model, takes the most time to train, and hence is executed at the last, having a huge buffer size.
A tweet enters the listeners process and it is added to the naive bayes, support vector machine, and deep neural network buffers. When a threshold for training the naive bayes model is reached (currently set to 100) , the tweets in the naive bayes buffer are trained. The training process ends, but the buffer is not cleared, yet. It is theoretically not possible to retrain a classifier for new classes, therefore this imposes a restriction. Another 100 tweets come into the listener process, and the naive bayes classifier is retrained for all the 200 tweets. The threshold kept for the support vector machine is 1000 tweets, therefore, when the number of tweets received reaches 1000, the support vector machine starts to train, and after its training process, it will be able to predict hashtags that were being predicted by the naive bayes model (since it already consisted of the tweets that were previously been trained by the naive bayes classifier). Therefore, the buffer of the naive bayes (the first 1000 tweets in the buffer) can now be deleted. Now, the naive bayes model will be trained for the next 900 tweets.
The same logic is extended for the deep neural network. The threshold set for the deep neural network model is 500000, because it takes a lot of time to train it, therefore adequate time should be given to it to train and be constantly used taking into the accuracy decrease of support vector machines when the number of classes increase. When the number of tweets increase to 500000, the tweets in the buffer for the deep neural network are used to train the neural network, whilst the naive bayes model and the support vector machine model continue to work as before, after the completion of the training process of the neural network, the buffer of the support vector machine (first 500000) and the buffer of the naive bayes (first 1000) are cleared, sice the deep neural network model is now able to predict the hashtags for the tweets contained in both naive bayes and svm model. Therefore, after clearing their respective buffers, they can work on newer tweets more efficiently.

VII. Results

This approach on building a recommender system to show users the possible hashtags that may be appropriate for the content of the tweet is highly time efficient. It can be used in the long run to give trending tweets providing social media admins an effective way to learn about trending hashtags that they could use with respect to their field, or in general, the current happenings about the world. Although this may be a very efficient method to give out recommendations, it consumes a lot of computational power, and must be run on a separate server, but it is definitely feasible in the real world.

We have observed that the naive bayes model gives us a better accuracy than the support vector machines for smaller number of classes as more data is required to be given to a support vector machine to fully it. The support vector machine outperforms the naive bayes at when the dataset is higher. This similar logic can be applied while training the deep neural network. The deep neural network requires a massive amount of data before it can be trained efficiently, but once trained it performs very well.

The results achieved in the project are above satisfactory, as all the models work well with each other, and the architecture of the overall machine learning model is complex but enables us to give results in real time, i.e., it helps us identify the events that are happening around the world in real time. The accuracy achieved by each model is quite satisfactory, as we are trying to classify over 5000 hashtags. If we were to randomly pick one hashtag, the probability of picking a correct recommendation would be 1/5000 = 0.0002, or 0.02%. But we are achieving an accuracy in the high 70s with the neural network. Similarly, we are able to achieve accuracies of 40% with the naive bayes model for about 50 classes, and about 50% for over 200 classes with the support vector machine.

VIII. Conclusion

Three models combined and executing in parallel gives more accuracy that running individual models. The buffering system is threaded, which takes care of the speed too. We need not wait indefinitely for a process to complete and waste time. This project has been made, taking into account its actual implementation in the real world. We started off with the idea of making a chrome extension out of the model but due to time constrains we were unable to do so. The chrome extension would actually make availabe, a set of hashtags, to the user, whenever he uses Twitter and writes a tweet.

IX. Contribution of Each Team Member

No project is complete without teamwork, and a sense of understanding between the team members. We proved to be an awesome team. However, on a broader level, the distribution of work is as follows :
Tanya Mehrotra -
i) Fetching data using Tweepy, the Twitter API.
ii) Preprocessing of data using NLTK
iii) Constructed the Naive Bayes Model and the Support Vector Machine Model.
Varun Ranganathan -
i) Came up with the idea to combine all three models and do something unique.
ii) Created the Deep Neural Network Model
iii) Combined all the three models and threaded them to work as one.

Acknowledgment

We would like to thank Dr. Gowri Srinivas, who gave us this wonderful platform to delve into the depths of Data Analytics. Amongst the vast pool of algorithms that we could have chosen, her diligence in teaching and our keen interest coupled together, assisted us in narrowing down to the most optimal solution. Secondly, we would like to acknowledge Hemkesh V, who actually motivated us when we approached him about the problems that we were facing, and ensured us that hard work would count always. Last, but not the least, we would like to thank Vishwas Satish, for being the cameraman for the video.

X. References

1) Kywe, Su, Tuan-Anh Hoang, Ee-Peng Lim, and Feida Zhu. On recommending hashtags in twitter networks. Social Infor- matics (2012): 337-350

2) Godin, Frderic, Viktor Slavkovikj, Wes- ley De Neve, Benjamin Schrauwen, and Rik Van de Walle. Using topic models for twit- ter hashtag recommendation. In Proceed-ings of the 22nd International Conference on World Wide Web, pp. 593-596. ACM, 2013

3) Popescu, Ana-Maria, Marco Pennacchiotti, and Deepa Paranjpe. ”Extracting events and event descriptions from twitter.” In Proceedings of the 20th international conference companion on World wide web, pp. 105-106. ACM, 2011.

4) Sriram, Bharath, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. ”Short text classification in twitter to improve information filtering.” In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval, pp. 841-842. ACM, 2010.