Academic Fora

FULL PAPER PROCEEDING
Multidisciplinary Studies
Available online at www.academicfora.com

Full Paper Proceeding ECBA -2016, Vol. 49- Issue.6, 1-4

ISBN 978-969-670-117-0

ECBA-16

REAL TIME INFORMATION CLASSIFICATION IN TWITTER USING STORM

AKHMEDOV KHUMOYUN[1], YUN CUI[2] & HANKU LEE[3*]

[1,2,3]Konkuk University

Keywords:
Classification
Storm, Statistics
Twitter
Information Retrieval
Big Data

Abstract. We are living in the digital era in which overly vast amount of information is generated almost constantly. The huge information hubs such as Twitter is one of the main sources of this diverse information space. However, with more than 500 million tweets sent per day as of 2015, identifying and classifying critical information when it first emerges on Twitter is a tremendous challenge. Our proposed work is an intention to tackle this challenge by discovering, extracting and qualifying into 5 generic categories from huge mess of public tweets in real-time. Going real time with such intention is not a trivial task. There has been extensive research on information retrieval on the topic. However, most existing works on classification of short text messages like tweets integrate every message with meta-information from external information sources such as Wikipedia and WordNet. Automatic text classification and hidden topic extraction techniques do well when there is meta-information or the context of the tweet is extended with knowledge extracted leveraging huge collections. But these approaches require online querying which is time consuming and unfit for real time applications. Hence, we propose slightly different intuitive approach to tackle this issue by leveraging $\chi 2$ statistical method as a feature extractor and Storm as a real-time data processing engine.

INTRODUCTION

Document classification is an example of Machine Learning (ML) in the form of Natural Language Processing (NLP). By classifying text, we are aiming to assign one or more classes or categories to a document, making it easier to manage and sort. This is especially useful for publishers, news sites, blogs or anyone who deals with a lot of content.

As mentioned earlier in the abstract page, automatic text classification and hidden topic extraction approaches perform well when there is meta-information or the context of the tweet is extended with knowledge retrieved from vast collections such as Wikipedia. However, these approaches require online querying which is time consuming and is not suitable for real time applications. When external features from the world knowledge is used to enhance the feature set, complex algorithms are required to carefully reduce overzealous features. These approaches eliminate the problem of data sparseness but create a new problem of the "Curse of dimensionality" (see "https://en.wikipedia.org/wiki/Curse_of_dimensionality"). Hence efficient ways are necessary to improve the accuracy of classification by using minimal set of features to represent the tweet. We use a statistical feature extraction method to determine the class labels and the set of features.

We classify incoming tweets into five generic categories – economics, politics, health, sports and miscellaneous messages. We believe that these categories are diverse and cover most of the topics. Experimental results using our proposed approach outperform the baseline Bag-Of-Words model in terms of accuracy and speed.

RELATED WORK

Most of the related work focuses on trying to eliminate the problem of data sparseness. One intuitive approach to achieve this is to inflate the short text with additional information to make it appear like a large document of text. Then, traditional classification or clustering algorithms can be applied to it. Some of the previous works (Sarah Zelikovitz, Haym Hirsh, 2004) processed the short text classification using the methods through unlabeled background knowledge to assess document similarity and unlabeled data in text classification problems, and (Mehran S., Timothy D. Heilman, 2005) proposed the document similarity

*Corresponding author: Hanku Lee
E-mail: hlee@konkuk.ac.kr

Peer-review under responsibility of the Scientific & Review committee of ECBA-2016.

computing method with web-based core function. Another work (D. Bollegala, Y. Matsuo, 2007; Sahami&Heilman, 2006; W. Yih, C. Meek, 2007) primarily focuses on integrating short text messages with Web search engines like Google, Bing to extract more information about the short text. More recent works (Phan & Nguyen, 2008; Hu, X. Sun, N. Zhang; Ramanthan & Gupta, 2007) do away with web searches and instead utilize data repositories such as Wikipedia. By integrating knowledge available within the Wikipedia, short text messages can be enhanced with more semantic knowledge. S. Ramanthan and Gupta made use of the user-defined categories and concepts extracted from Wikipedia and experimental results show significant improvement in accuracy. This approach however will not capture the up-to-date information and is especially unsuitable when the input data is highly volatile in its theme like news feeds. Banerjee et al also mentions that usage of additional Wikipedia 8 concepts (other than titles) did not offer any significant improvements in performance of various clustering algorithms. Phan and Nguyen (2008) not only uses the explicit user defined categories in Wikipedia but extracts hidden topic of Wikipedia articles to gain more knowledge. Although it eliminates the problem of data sparseness, it is very time consuming and there is a need to understand what concepts of Wikipedia are useful to extract as mentioned in (S. Ramanthan and Gupta, 2007).

One of the biggest challenges in almost all the related work so far is that by enhancing the feature set by using external knowledge; a new problem creeps in (the Curse of Dimensionality). When the feature set becomes too large, data becomes difficult to visualize and the basis for classification or clustering is lost. Hence, there is a need to effectively reduce features and the feature size to an optimal value.

In conclusion, related works on short text messages in recent times have primarily focused on eliminating the problem of data sparseness by using external web sources like Wikipedia, WordNet etc. Querying such sources online poses the problem of longer time whereas using a snap shot of such data sources has the problem of out dated information. In this work, we propose the use of a small feature set to classify Twitter messages in combination with Naïve Bayes algorithm. Initially, we classify the Twitter messages into diverse pre-chosen classes like Economics, Politics, Health, Sports and miscellaneous.

The rest of the paper is organized as follows. First, we provide an overview of Apache Storm. After that, we give a brief overview of data collection phase. Then, we discuss our proposed work in detail. The next chapter contains details about the experimental results and comparison of the proposed work with some baseline algorithms. Finally, we conclude with the future work.

OVERVIEW OF APACHE STORM

Apache Storm, a distributed computation framework, created by Nathan Marz, adds reliable real-time data processing capabilities to Apache Hadoop. It is fast, scalable, reliable and can be programmed using a variety of programming languages. Its architecture consists of three primary node sets:

Nimbus Node

This is the master node. It uploads the computation to be performed in the cluster, launches worker nodes and even re-assigns worker nodes in case of failure. There is only one master node in a cluster.

Zookeeper Nodes

These nodes are assigned on every slave machine. The basic function of the zookeeper nodes is to keep a check on the processing happening on the worker nodes. The nimbus communicates with the worker nodes through the zookeeper.

Supervisor Nodes

There nodes, assigned on every slave machine, start and stop workers according to commands from the nimbus. There can be multiple worker nodes on a single slave machine. A key abstraction in Storm is the topology which is in fact, the program that keeps running in the Storm cluster. A visual representation of the topology is a network of the spout and bolts that Storm employs to perform its computation. A spout is an input stream that generates input tuples. Spouts are the source of data in a Storm cluster. In order to receive real-time data, a spout can be configured with an API or a queuing framework like Kafka. The spout sends data to bolts. These bolts are where the actual processing takes place and a cluster may have multiple bolts for the various processing steps required to achieve the desired result. Bolts can pass data further on to another bolt or to a location of data storage.

The reason we chose Storm is that, firstly, it provides very simple API using high level components like Spout and Bolts, and also it is faster compared to other similar techs.

DATA COLLECTION

Twitter offers three ways to access its data and these are:

Twitter Search API

Twitter Search API helps to access a data set that exists from tweets previously written. In the Search API, users ask for tweets that match a search criteria or even a part of it. The criteria could be keywords, usernames or locations. But there is a limit to the number of tweets that can be accessed. For an individual user, the maximum number of tweets that can be received is the last 3,200 tweets, regardless of the query criteria. With a specific keyword, Twitter only polls the last 5,000 tweets per keyword.

Twitter Streaming API

Twitter Streaming API is a push of data as tweets happen in near real-time unlike the Twitter Search API. With Twitter Streaming API, users register a set of criteria (keywords, usernames,

locations, named places, etc.) and as tweets match the criteria, they are pushed directly to the user. This API is free of cost although the percentage of total tweets users receive with the Twitter Streaming API varies heavily based on the criteria users request and also on the traffic. Therefore, use of this API cannot be relied upon to generate a large number of tweets for real-time processing.

Twitter Firehose

Twitter Firehose guarantees delivery of 100% of the tweets that match the criteria. It is very similar to the Twitter Streaming API as it pushes data to end users in near real-time. But the difference lies in the cost. Twitter Firehose is not free of cost, unlike the Streaming API.

Implementation

Storm spout is linked to the Twitter API and is responsible for the streaming of tweets into the Storm cluster. The spout does not perform any processing on the data. It simply streams it. These tweets are then sent by the spout to the bolt in the cluster so that they can be processed.

PROPOSED SYSTEM

Here we present detailed walkthrough of the workflow and dataflow of the proposed system. First we give step-by-step overview of the data processing workflow of the system which contains 3 steps: Preprocessing, Feature extraction and Classification. Then, we present dataflow of the system in terms of spout and bolts.

Workflow of the System

Workflow of the proposed system consists of three phases: Preprocessing, Feature extraction and Classification.

Preprocessing

This phase is mainly concerned with data cleaning because tweets are by nature very noisy which contains items like slangs, uncomplete forms of words, emoticons, etc. Hence, first we remove non-English words because our work focuses on English tweets. Then, we perform case normalization in which all incoming tweets are converted into lowercase. After that, we perform stemming, the procedure of decreasing relevant tokens into a single type of token. This procedure contains the recognition and elimination of suffixes, prefixes and unsuitable pluralization. After preprocessing the tweets we perform feature extraction based on statistical method.

Feature Extraction

χ^2 -based feature extraction. The feature extraction algorithm based on statistics is as follows:

Input: Training text (tweet) set T_{tset}

Output: Category feature set F_{set}

1) For every tweet of the training set, segmenting and part of speech (POS) tagging, and then marking the set as Tl_{set}

2) Removing the stop words and the words with stop POS, and counting the word frequency and the word file frequency. Tl_{set}

3) For given category, computing its words weight using the following method:

4) Counting the total number N of training tweets

5) Computing A as the number of tweets in which term w appears in the given category Ci

6) Computing B as the number of tweets in which term w appears in other classes excluding Ci

7) Computing C as the number of tweets in which term w disappears in the category Ci

8) Computing D as the number of tweets in which term w disappears in other categories excluding Ci

9) Computing the word weight of w in the given category using the following expression:

$$W_{weight} = N \times (A \times D - B \times C)^2 / ((A+B) \times (A+C) \times (B+D))$$

10) Turn to 3) until all words of this category are completely calculated.

11) Order the words by the weight descending

12) Selecting the top M words as this category features, M can be 50, 100 or 200, etc.

13) End

Classification

In the final classification phase we leveraged several ML classification algorithms such Naïve Bayesian, SVM (Support Vector Machines) and SMO (Sequential Minimal Optimization). We conducted experimental comparison of the above mentioned algorithms in terms of efficiency and we will provide detailed report in the next section. Here, we did not include each algorithm's details for brevity since it will consume a lot of space. But we included links to the sources in the reference page in which one can find details of each algorithm.

Dataflow of the System

Here, we present dataflow of the system in terms of topology of Storm spout and bolts. First, CrawlerSpout, which is responsible for retrieving live public tweets, will connect to Twitter's Streaming API using twitter4j library. Then, DataCleanerBolt pulls tweets from the spout for efficient data cleaning from where then cleaned tweets are pulled by PreprocessingBolt which is responsible for preprocessing step defined in workflow of the system. After this, step FeatureExtractorTFIDFBolt, FeatureExtractorBOWBolt and FeatureExtractorSTATBolt bolts as names suggests will extract features from tweets according to their definition. In this step, main importance is given to FeatureExtractorSTATBolt since it is the implementation of our proposed approach, χ^2 statistical method for feature extraction.

Finally, in the last step of our storm topology, implementations of three classification algorithms, Naïve Bayesian, SVM and SMO, NBClassifierBolt, SVMClassifierBolt and SMOClassifierBolt will classify tweets coming from feature extractor bolts and write the results to filesystem for further analysis. The high level overview of the described topology can be seen in Figure 1, which is provided at the end of the paper.

EXPERIMENTAL RESULTS

In this chapter, we present the experimental results for techniques described in the earlier chapter. All experiments were run using available implementation of Weka. Three classification algorithms, namely Naïve Bayes, SVM and Sequential Minimal Optimization (SMO) were used on the training data. Experimental results are based on 5-fold cross validation of the data.

The features used for each of these experiments are as follows:

- **TF-IDF**: Term frequency-invert document frequency
- **BOW**: Bag-Of-Words is chosen as the baseline
- **Statistical method**: our proposed method

We ran our Storm topology in 3-node cluster for 1 hour. The amount of processed tweets in this period was over 10, 000 excluding non-English tweets. The distribution of tweets per category is shown in Figure 2.

In the Figure 3, it can be seen that our proposed statistical method performs the best amongst any other chosen feature set for all the three algorithms. BOW has the least accuracy among all other feature sets.

CONCLUSIONS AND FUTURE WORK

The work described in this paper is a step towards efficient classification of short text messages. Short text messages are harder to classify than larger corpus of text. This is primarily because there are few word occurrences and hence it is difficult to capture the semantics of such messages. Hence, traditional approaches like "Bag-Of-Words" and TF-IDF when applied to classify short texts do not perform as well as expected.

We have proposed a framework to classify Twitter messages which serve as an excellent candidate for short text messages because of their 140 character limit. In this framework, we leveraged several well-known classification algorithms namely, Naïve Bayesian, SVM and SMO, using Apache Storm.

In this work, we mainly focused on tweets from Twitter for classification, but short messages are not limited to just tweets. Hence, we have a plan to improve our framework to work with other types of short messages like blog comments, news comments and instant messages.

REFERENCES

Bollegala, D., Matsuo, Y., & Ishizuka, M. (2007). Measuring semantic similarity between words using web search engines. *www*, *7*, 757-766.

Sahami, M., & Heilman, T. D. (2006, May). A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th international conference on World Wide Web* (pp. 377-386). AcM.

Yih, W. T., & Meek, C. (2007, July). Improving similarity measures for short segments of text. In *AAAI* (Vol. 7, No. 7, pp. 1489-1494).

Phan, X. H., Nguyen, L. M., & Horiguchi, S. (2008, April). Learning to classify short and sparse text & web with hidden topics from large-scale data collections. In *Proceedings of the 17th international conference on World Wide Web* (pp. 91-100). ACM.

Hu, X., Sun, N., Zhang, C., & Chua, T. S. (2009, November). Exploiting internal and external semantics for the clustering of short texts using world knowledge. In *Proceedings of the 18th ACM conference on Information and knowledge management* (pp. 919-928). ACM.

Banerjee, S., Ramanathan, K., & Gupta, A. (2007, July). Clustering short texts using wikipedia. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 787-788). ACM.

Zelikovitz, S., & Hirsh, H. (2000, June). Improving short text classification using unlabeled background knowledge to assess document similarity. In*Proceedings of the seventeenth international conference on machine learning*(Vol. 2000, pp. 1183-1190).

Zelikovitz, S., & Hirsh, H. (2004). Transductive LSI for Short Text Classification Problems. In *FLAIRS conference* (pp. 556-561).

Anonymous. (n.d). *Naïve Bayesian.* Reteved from https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Anonymous. (n.d). *Support vector machine.* Retrieved from https://en.wikipedia.org/wiki/Support_vector_machine

Anonymous. (n.d). *Sequential minimal optimization* Retrieved from https://en.wikipedia.org/wiki/Sequential_minimal_optimization

Sahami, M., & Heilman, T. D. (2006, May). A web-based kernel function for measuring the similarity of short text snippets. In *Proceedings of the 15th international conference on World Wide Web* (pp. 377-386). AcM.

Anonymous. (n.d). *Curse of Dimensionality.* Retrieved from https://en.wikipedia.org/wiki/Curse_of_dimensionality.