COMPARING RESULTS OF SENTIMENT ANALYSIS USING NAIVE BAYES AND SUPPORT VECTOR MACHINE IN DISTRIBUTED APACHE SPARK ENVIRONMENT

Tomasz Szandala

Department of Computer Engineering
Wrocław University of Technology, Wrocław, Poland

ABSTRACT

Short messages like those on Twitter or Facebook has become a very popular opinions sharing tool among Internet users. Therefore micro blogging web-sites are nowadays rich sources of data for opinion mining and sentiment analysis. However it is challenging because of the limited contextual information that they normally contains. Furthermore the greatest benefit can be achieved by collecting sentiment class in real time - when the post is published, in order to react as soon as possible. Nevertheless, most existing solutions are limited in centralized environments only. thus, they can only process at most a few thousand tweets. Such a sample, is not representative to define the sentiment polarity towards a topic due to the massive number of tweets published daily. Sample analysis has been performed using Machine Learning methodologies alongside with Natural Language Processing techniques and utilizes Apache Spark's Machine learning library, MLlib, on a labelled (positive/negative) corpus containing 4234 tweets regarding Presidential Election in USA in 2016. The analysis has been completed using distributed Apache Spark environment with simulated stream of data from Kafka database.

KEYWORDS

Apache Spark, natural language processing, sentiment analysis

1. INTRODUCTION

Sentiment analysis is a group of methods, techniques, and tools about detecting and extracting subjective information, such as opinion and attitudes, from any text or voice source. [1] Traditionally, sentiment analysis has been about opinion polarity, i.e. whether someone has positive, neutral, or negative attitude towards something or someone. The best applications for attitude recognizing include market survey about new products or political parties/candidates pre-voting support.

Any research appears more attractive when its results can be compared to real life results in whole population. This is why this paper is being focused on Presidential Election in USA in 2016, because scientifically obtained results can be compared to actual results that gave White House to Donald Trump.

David C. Wyld et al. (Eds) : AI, CSTY, NMOCT - 2018
pp. 15–21, 2018. © CS & IT-CSCP 2018 DOI : 10.5121/csit.2018.81402

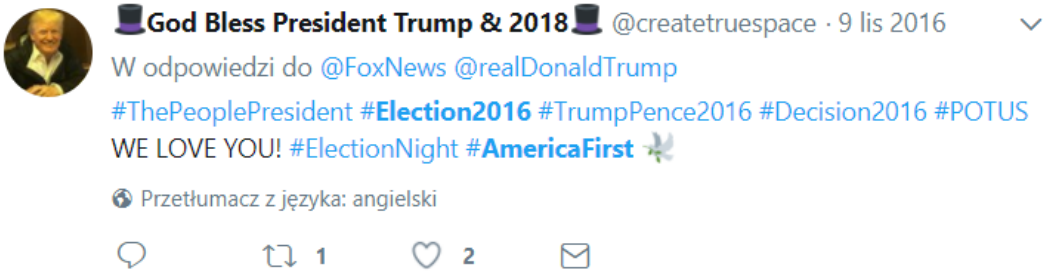
Twitter company has to processes on average 8000 tweets per second (and over 12000 in peak hours) [2] before publishing them for public access. They analyze all data with this extreme fast rate, to ensure that every tweet is following agreement policy and restricted words are filtered out from the messages. All this analyzing process has to be done in real time to avoid delays in publishing tweets live. To analyze such huge data it is required to use some kind of analysis tool. This paper chooses an open source tool Apache Spark. Spark is a cluster computing system from Apache Foundation focused on analyzing any data in parallel[3].

Spark MLlib (Machine Learning Library) is a distributed machine learning framework on top of Spark's Core that, due in large part to the distributed memory based Spark architecture, is as much as nine times as fast as the disk-based implementation used by Apache Mahout (according to benchmarks done by the MLlib developers against the Alternating Least Squares (ALS) implementations, and before Mahout itself gained a Spark interface), and scales better than Vowpal Wabbit[4]. Many common machine learning and statistical algorithms have been implemented and are shipped with MLlib which simplifies large scale machine learning activities.

And last, but not least comes the Apache Kafka database. In this paper it is used to simulate stream of tweets. Even tho Twitter is sharing quite versatile API for fetching specific tweets, the environment in which research has been performer, suffered from unstable network connection which forced emergency measures.

2. TWEET ANALYSIS

A typical tweet data consists of up to 140 characters (as of 7 November 2017 the amount has been doubled). Apart from main text the tweet usually have emoticons and hashtags.



The tweets were chosen by fetching all tweets with hashtags "#Election2016", "#TrumpPence2016" "#Clinton2016", "#POTUS" and few more recommended by Twitter itself. Of course all analyzed tweets came from between 1 June 2016 and 10 November 2016.

Tweets has been at first divided into two groups: Trump or Clinton. Assignment has been made by recognizing any reference to specific candidate like name, surname, vice president surname or motto, like "#AmericaFirst" which can be indisputably connected to one of them. If the tweet contained references to both groups or to none of them it has been rejected from further processing In the next step all messages has been preprocessed. I have converted all of our text to lower-case, removed hashtags from the start and end of the message. I have taken an assumption that surrounding hashtags does not carry any sentiment value, while inner ones (e.g. "#proud") can be valuable in analysis. Since Twitter is very casual, people often include multiples of the same letters in a word, such as “Weeeee wooooooon.” To handle these cases, repeating characters has been reduced to no more than two of the same consecutive letters. All URLs and user names were also discarded since URLs and user names are rather not related to the emotion expressed in the tweet. Furthermore, a standard list of common words that do not express sentiment has been

skipped. The list includes words like “be,” “at,” “the,” etc, which might only slow down analysis performance.

Emoticons were also taken into consideration and were translated into corresponding words (e.g. :-) into "happy"). For the final model for analysis I have chosen simple bag-of-words model [5,6]. It is a simplifying representation used in natural language processing and information retrieval. Also known as the vector space model. In this model, a text (such as a tweet) is represented as the bag (multiset) of its words, disregarding grammar and even word order but keeping multiplicity. All other models, even tf-idf, which is considered to be more accurate, has no point in tweets analyzing since messages are too short to see full benefit from them.

3. TOOLS FOR DISTRIBUTED DATA ANALYSIS

3.1. General Format, Page Layout and Margins

As I have mentioned in the introduction Twitter has to solve few thousands messages per second. Since adding any advanced analysis to text, like sentiment, may introduce delays in receiving verdict there is a need for an efficient tool. This problem was noticed by Matei Zaharia who proposed in his PhD [7] a distributed platform for parallel data analysis. Apache Spark is a distributed computing framework which performs operations on the working data sets, which are called resilient distributed datasets (RDDs)[3].

Apache Spark’s programming model is based on processing those RDD objects in a form of acyclic data flow, using a set of operators called transformations. Such model supports greatly operations derived from functional programming like mapping, filtering or reducing. Functional paradigms allows to easily split the computation that would be normally performed in a single thread way over multiple cluster nodes. To get the most benefits from this parallelism, the data should be first placed in distributed file system like Hadoop Distributed File System. Even though Apache Spark is written in Scala (functional version of Java) and is run on Java Virtual Machine, it also natively supports Python scripting for data analysis.

3.2. Apache Kafka database

Apache Kafka database is an open-source stream processing platform developed by the Apache Software Foundation written in Scala. The main features of the project is to provide a unified, high-throughput, low-latency platform for handling real-time data feeds. Its storage layer is essentially a "massively scalable pub/sub message queue architected as a distributed transaction log,"[8] making it extremely valuable for enterprise infrastructures to process streaming and/or distributed data.

4. CLASSIFICATION METHODS

4.1. Naive Bayes classifier

One of simplest classification methods is naive Bayes classifier[9,10]. Naive Bayes is a simple multiclass classification algorithm with the assumption of independence between every pair of features. Naive Bayes can be trained very efficiently even on small size of training data. Within a single pass to the training data, it computes the conditional probability distribution of each feature given label, and then it applies Bayes’ theorem to compute the conditional probability distribution of label given an observation and use it for final prediction in validating set.

4.2. Linear Support Vector Machine

Since we are considering only two classes of possible verdict - positive or negative, we can choose binary classifier for sentiment analysis. Support Vector Machines (SVMs, also support vector networks) is supervised learning model with associated learning algorithms that analyze data used for classification and regression analysis [10, 11]. Given a set of training examples, each marked as belonging to one or the other of two categories.

$$(x_1 ,y_1),(x_2 ,y_2),. . . (x_n ,y_n) \quad (1)$$

Where (eq. 1):

x_i are the vectors of features and y_1 is on of binary class label: -1 or +1. We want to find the "maximum-margin hyperplane" that divides the group of points for which y=-1 from the group of points for which y=1, which is defined so that the distance between the hyperplane and the nearest point from either group is maximized. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.

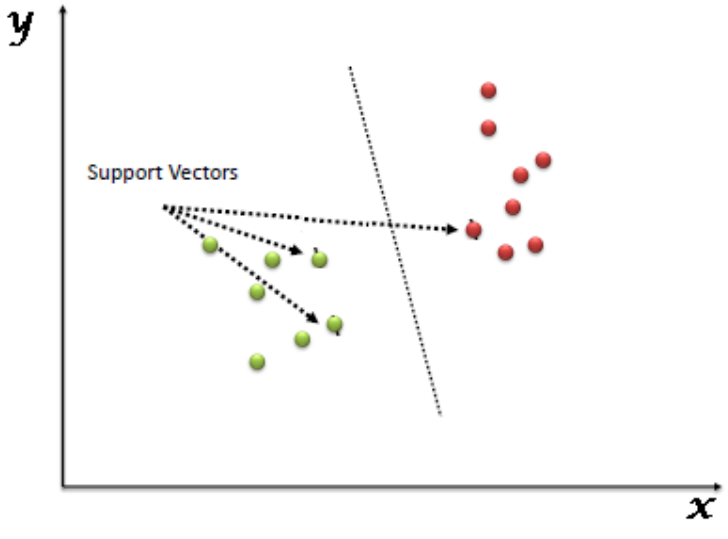


Fig. 1

Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line)

5. THE EXPERIMENT

5.1. Training

The source tweets were shuffled and splitted up into training and test sets in-order to validate the trained model. The division for this work is carried out in the ratio of 3:1. This means: 3176 of the messages are used for training and rest of the data is left for later testing. It is also worth to mention that around 70% of tweets were negative. At this point the target of the tweet (Trump or Clinton) were not taken into consideration.

5.2. Validating

The measure of classifier performance was accuracy, the amount of correctly recognized messages divided by quantity of all messages taken into consideration. The process of training and validating has been repeated five times and the most promising pair of classifiers (the one with highest accuracy) will be taken into classifying real data.

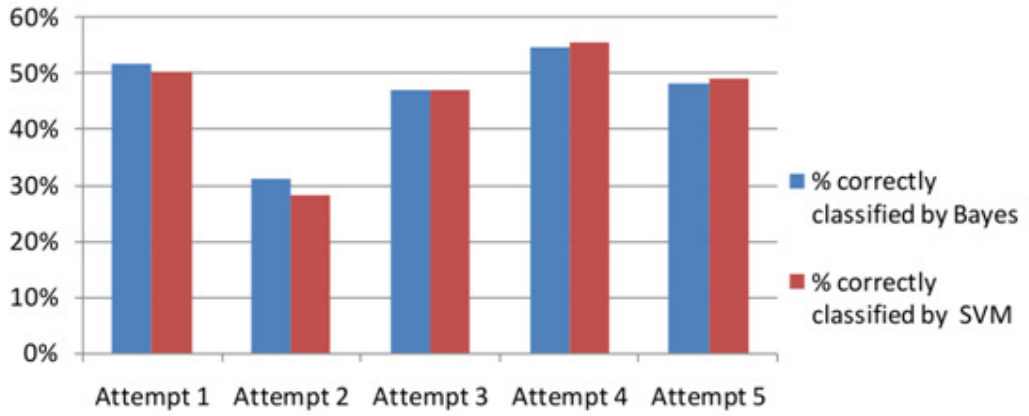


Fig. 2 Comparison of correctly classified tweets between attempts

The results (fig. 2.) shows that attempt fourth is the most accurate one for both classifiers. Bayes classifier was 54,7% time right and SVM achieved 55,6% accuracy. However the second attempt shows distinctive results. After closer look I have noticed that after shuffling almost all positively labeled messages fell into validating set and since training set was extremely unbalanced we can see biased results.

5.3. The real data

Last step of the experiment was to use the best fitted classifiers for analyzing sentiment of the real data. At first the 40 000 tweets with hash tag, associated with elections, were downloaded and stored into Kafka database. 20 000 regarding Trump, 20 000 about Clinton. Afterwards the analysis has been performed using previously trained classifiers and Apache Spark streaming on single node. In the next iterations I have added one and later second slave node to Apache Spark.

Table 1 Sentiment analysis on randomly chosen 20 000 tweets per candidate

	Trump positive	Trump negative	Clinton positive	Clinton negative
Bayes	9 595	10 405	7 378	12 622
SVM	8 998	11 002	6 841	13 159

Since real tweets have no label about its sentiment analysis we cannot judge classification accuracy. Nevertheless dominating negative sentiment (table 1.) may be prove for common belief that American citizens were voting for candidate they disliked less[12].

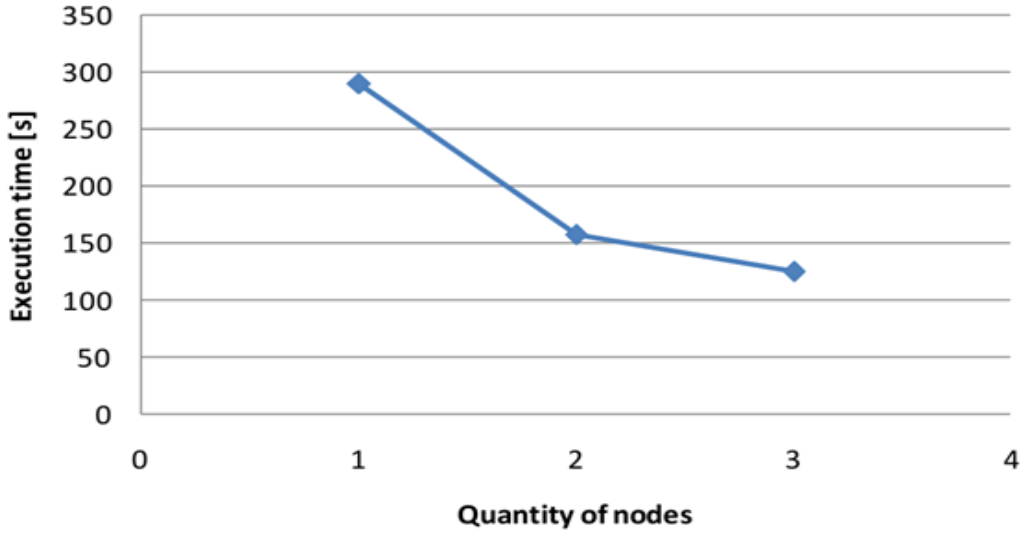


Fig. 3 Comparison of time of execution on 1, 2, 3 single thread Apache Spark nodes

Introducing parallel processing provides meaningful speed up in data processing as we can see in fig. 3. There is around 45% improvement on execution time if we add second node and close to 60% in total if we split our computation on 3 executors.

6. CONCLUSIONS

Papers in this format must not exceed twenty (20) pages in length. Papers should be submitted to the secretary AIRCC. Papers for initial consideration may be submitted in either .doc or .pdf format. Final, camera-ready versions should take into account referees' suggested amendments.

6.1. Naive Bayes vs SVM

While comparing accuracy of Naive Bayes classification and Support Vector Machine we do not see significant difference between them. Only the most important drawback of SVM is that we can compare only two classes with one such classifier. If we choose to set sentiment as positive or negative there will be no issue in it. But if we add for example third class "neutral" we would have to consider 3 instances of SVM classifier: each for each pair of classes and then use voting mechanism to choose the best fitting one.

6.2. Distributed processing in Apache Spark.

Introducing parallel processing feature from Apache Spark we can notice significant improvement in time of execution as we add more nodes. Furthermore Apache Spark ensures continuity of work even if one of the nodes disconnects. It rebalances to load to other working node(s) as long as at least one is up. At worst we can lose only data that was being processed at the moment of downfall.

6.3. Application in real data

As mentioned above: we can conclude that negative opinions dominated in 2016 Elections. Unfortunately twitter analysis show clearly that Trump should have easily won the elections. While he indeed won the election it happened only due to American specific electoral voting

system. In raw numbers Donald Trump would have lost to Hillary Clinton 46% to 48% votes. This means that twitter population is not fully representative and even if sentiment analysis would be perfected we cannot rely only on it.

REFERENCES

[1] M. V. Mantyla, D. Graziotin, M. Kuutila: The evolution of sentiment analysis A review of research topics, venues, and top cited papers, Computer Science Review, Volume 27, 2018.

[2] [online] Available: http://www.internetlivestats.com/twitter-statistics.

[3] Zaharia, M., Chowdhury, M., et al.: Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing. USENIX Symp. Networked Systems Design and Implementation

[4] Sparks, E.; Talwalkar: Spark Meetup: MLbase, Distributed Machine Learning with Spark. slideshare.net. Spark User Meetup, San Francisco, California , 2014.

[5] Youngjoong K.:"A study of term weighting schemes using class information for text classification SIGIR'12. ACM, 2012.

[6] Mikolov T., Chen K., Corrado G., Dean J.: Efficient Estimation of Word Representations in Vector Space 2013.

[7] Zaharia, M.: An Architecture for Fast and General Data Processing on Large Clusters. University of California, Berkeley. 2015.

[8] Mouzakitis E.: Monitoring Kafka performance metrics, www.datadoghq.com/blog, 2016.

[9] Russell S., Norvig, P.: Artificial Intelligence: A Modern Approach (2nd ed.). 2003

[10] Rish, I.: An empirical study of the na ive Bayes classifier. IJCAI Workshop on Empirical Methods in AI, 2001.

[11] Cortes C., Vapnik V.: Support-vector networks. Machine Learning. 20 (3), 1995.

[12] Long H.: Voters say this is the ultimate 'lesser of two evils' election, 2016.

AUTHORS

Tomasz Szandala – PhD candidate From Wroclaw, Poland, involved in multiple projects in the field A.I. Apart from university a DevOps engineer at NOKIA.