An Apache Spark Implementation for Graph-based Hashtag Sentiment Classification on Twitter

Elias Dritsas
Department of Computer Engineering and Informatics,
University of Patras,
Patras, Greece.
eldritsas@gmail.com

Ioannis E. Livieris
Department of Computer Engineering & Informatics,
Technological Educational Institute of Western Greece,
Antirrio, Greece.
livieris@teiwest.gr

Konstantinos Giotopoulos
Department of Business Administration,
Technological Educational Institute of Western Greece,
Patras, Greece.
kgiotop@ceid.upatras.gr

Leonidas Theodorakopoulos
Department of Business Administration,
Technological Educational Institute of Western Greece,
Patras, Greece.
theodorakopoulospl@gmail.com

ABSTRACT

Sentiment Analysis has been extensively investigated in recent years as a method of human emotions' classification to specific events, products, services etc. It is considered as a very important problem, especially for organizations or companies who want to know the consumers' view about their products and services. In combination with the evolution of social media, it has been established as an interesting domain of research. Through social media, people tend to express their opinions or feelings, such as happiness or sadness on a daily basis. Thus, the vast amount of available data has made the existing solutions inappropriate and the need for automated analysis methods is imperative. In this paper, we examine sentiment polarity analysis on Twitter data in a distributed environment, known as Apache Spark. More specifically, inspired from [23], we propose three classification algorithms for tweet level sentiment analysis in Spark due to its suitability for Big Data processing against its predecessors, MapReduce and Hadoop.

CCS CONCEPTS

• **Information systems → Collaborative and social computing systems and tools**; **Sentiment analysis**; • **Computing methodologies → Supervised learning**; • **Theory of computation →** *MapReduce algorithms*;

KEYWORDS

Apache Spark; Big Data; Classification; Microblogging; Sentiment Analysis; Supervised Machine Learning; Twitter.

ACM Reference Format:
Elias Dritsas, Ioannis E. Livieris, Konstantinos Giotopoulos, and Leonidas Theodorakopoulos. 2018. An Apache Spark Implementation for Graph-based Hashtag Sentiment Classification on Twitter. In *22nd Pan-Hellenic Conference on Informatics (PCI '18), November 29-December 1, 2018, Athens, Greece.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3291533.3291552

PCI '18, November 29-December 1, 2018, Athens, Greece

ACM ISBN 978-1-4503-6610-6/18/11…$15.00
https://doi.org/10.1145/3291533.3291552

1 INTRODUCTION

Nowadays, the vast evolution of Internet has radically changed the ways of communication, information and interactions between people. Internet users are no longer passive information receivers but in contrast, they participate in social networks and have the opportunity to discuss with others by exchanging views and ideas. Specifically, users tend to disseminate information, through short 140-character messages called "tweets" or follow other users in order to receive their status updates. Twitter constitutes a wide spread instant messaging platform which people use in order to get informed about world news, technological advancements, etc. Inevitably, a variety of opinion clusters that contain rich sentiment information, is formed.

The rapid growth of Internet has significantly increased the data volume and made the processing with the traditional ways very difficult. Therefore, there is an increasing need to move into cloud computing technologies since they provide tools and infrastructure for creating high scalable solutions and managing the input data in a distributed way among multiple servers.

The public opinion around various issues is reflected through questionnaires and surveys. As more users post reviews for products or services they use, the social networking platforms are becoming important sources of information, which is for the first time recorded directly in electronic form. The need for analysis and recovery of the produced volume of information in an automated way, paves the way to Sentiment Analysis [13]. According to [15], Sentiment Analysis is one of the most simple problems in Natural Language Processing; the computing system does not need to completely perceive the semantic of each proposition but to detect the whole attitude of the author and in following classify it according to its polarity. However, the polarity detection problem seems to be difficult even for the humans. Sentiment Analysis can be applied in various levels depending on text size and the desired resolution of exported information:

(1) **Document Level:** it is assumed that in each document, an opinion on a particular subject or topic is expressed.

(2) **Sentence Level:** it separates each document proposals assuming that each proposal expresses a single view.
(3) **Attribute Level:** it separates each document or proposal to phrases which they refer to an entity as a whole or separately for each feature with a different feeling.

In the context of this work, we utilize *hashtags* and *emoticons* as sentiment labels to perform classification of diverse sentiment types. Hashtags are a convention for adding additional context and metadata and are extensively utilized in tweets [23]. They are used to provide categorization of a message and/or highlight of a topic and they enhance the searching of tweets that refer to a common subject. Both, hashtags and emoticons, provide a fine-grained sentiment learning at tweet level which makes them suitable to be leveraged for opinion mining.

Previous works regarding emotional content are the ones presented in [10–12], in which the authors presented various approaches for the automatic analysis of tweets and the recognition of the emotional content of each tweet based on Ekman emotion model, where the existence of one or more out of the six basic human emotions (Anger, Disgust, Fear, Joy, Sadness and Surprise) is specified.

In this paper, Sentiment Analysis on Twitter data based on [23] is performed. Our proposal is three classifiers, namely Naive Bayes, Logistic Regression and Decision Trees, which we implemented in Apache Spark, a prominent, distributed environment appropriate for processing large-scale data. The algorithm uses a set of categorized tweets to train the classifiers and in following it classifies the tweets on three categories, namely positive, negative or neutral.

The rest of the paper is organized as follows: in Section 2 we discuss related work as well as Spark framework is presented while in Section 3, the Sentiment Analysis Classification Framework is introduced. Section 4 presents the datasets for validating our framework. Moreover, Section 5 presents the evaluation experiments conducted and the results gathered. Ultimately, Section 6 presents conclusions and draws directions for future work.

2 RELATED WORK

2.1 Sentiment Analysis and Classification Models

In the last decade, there has been an increasing interest in Sentiment Analysis [19] as well as emotional models. This is mainly due to the recent growth of data available in the World Wide Web, especially those which reflect people's opinions, experiences and feelings [18]. Early opinion mining studies focus on document level sentiment analysis concerning movie or product reviews [7, 26] and posts published on web pages or blogs [25]. In recent years, thanks to the development of social media, the interest of academic community and industry has turned to the processing and analysis of massive data. This resulted in important research studies which solely focus on data from social networks. Below is an overview of the main studies that have been conducted regarding Twitter data, divided into Supervised [22] and non-Supervised Machine Learning techniques, where the former achieve better precision than the latter ones.

2.1.1 Supervised Machine Learning Approaches. Study [6] is among the first ones conducted in Sentiment Analysis on Twitter data. Authors adopt binary sentiment classification problem by describing tweets as positive or negative. They apply a distant supervised technique in order to train a supervised machine learning classifier and in following compare the algorithms of Naive Bayes [13, 20], Maximum Entropy and Support Vector Machines. The final training set includes tweets for subjectivity detection and positive or negative tweets for polarity classification. Another interesting approach is described in [3] for evaluating the effect of small length tweets to the usual techniques for supervised machine learning. Authors collected tweets from the ten most popular topics in five categories (entertainment, products & services, sports, news and companies) by creating a set of manually categorized standards. During the preprocessing step, they replace usernames, hashtags and hyperlinks with pre-built keywords. As text representation attributes, monograms, digrams, trigrams, POS tags as well as POS v-letters are used.

A two-phase classifier is proposed in [2]; in the first phase, the tweets are categorized as subjective (have emotion) or objective (neutral) and then are subjectively distinguished into positive or negative. For the construction of the training set, noisy emoticons are used as labels and three emotion detection tools (Twendz 2, Twitter Sentiment 3 and TweetFeel 2) are put to use. The authors in [4] automatically classify the data set of [16] by using as categorization indexes (noisy labels) consisting of 50 hashtags and 15 emoticons. The set of features includes words, v-letters $(2-5)$, the length of each tweet, the plurality of punctuation, exclamation points, question marks, capital letters and words, along with the existence of words with high frequency.

In addition, a three-stage technique in [8] considers the emotion analysis on specific target dependent topics. Similar to [2], the authors initially classified tweets as subjective and objective and then (as second phase), with the use of two separate Support Vector Machines (SVM) classifiers by linear kernel function, the subjective tweets are classified as positive or negative. They claim that conventional techniques such as the ones proposed in [6, 24] are not sufficient as all the features are target independent. In the third phase, they propose a graph-based method in order to increase the accuracy. They observe that their approach leads to better performance against the one proposed in [2], as their method is mainly based on lexical features instead of more abstract ones.

A method for corpus collection and in following for building a sentiment classifier that is able to determine positive, negative and neutral sentiments is proposed in [17]. The classifier is based on the multinomial Naive Bayes classifier that uses n-gram and POS-tags as features considering conditional independence of n-gram features and POS information. Their experimental evaluations on a set of real microblogging posts prove that their technique is efficient and performs better than previously proposed methods.

2.1.2 Non-Supervised Machine Learning Works. The connection of the polls with Sentiment Analysis on tweets that referred to US President Barrack Obama is considered in [16]. Specifically, through Twitter API, 1 billion tweets posted during the period from 2008 to 2009 without checking the demographic characteristics of the authors and the writing language, are collected. Authors classify each tweet by measuring if it contains more positive or negative words,

while seeking the polarity of each word in the dictionary of MPQA emotions [24].

Furthermore, authors in [5] consider the predictability of the final result applying popular techniques to tweets regarding the election of 2010 US Representatives. These techniques are based on the previous method in [16], using the MPQA dictionary [24] and introduce some changes so that the overall model suits to the nature and characteristics of each electoral system. Taking into account tweets that contain the names of rival candidates, they do not allow a tweet to have two opposite polarities simultaneously. Another non-supervised hybrid method is the one proposed in [14], which deals also with tweets experimenting on large body texts and dictionaries in order to determine the semantic orientation of the text terms. The proposed method of calculating the emotion value takes into account both the polarities of the dictionary and the number of emoticons, repeated letters, exclamation marks as well as capital letters, which are features commonly used by supervised techniques.

Moreover, a model based on entity level regarding data collected from Twitter is proposed in [25]. This model applies pre-processing to the corresponding data set by deleting duplicates, removing usernames and hyper-links, replacing abbreviations to their normal form and finally recognizing grammatical terms of individual messages (POS tagging). Then, it calculates the emotional value of each term based on its similarity with emotions from dictionary words and solves the simple reports by assigning pronouns to the nearest entity of the text. The following step allows a binary SVM classifier, with standards resulted from the above non-supervised procedure, which classifies tweets in their final classes, to be educated.

In this work, we adopted the approach introduced in [23], which belongs to supervised machine learning techniques, and tried to implement it in a distributed environment, ideal for big data management. More specifically, the emotion analysis is proposed to be implemented in two phases, which are the tweet and hashtag level. In tweet level, a two-phase SVM classifier, like [2], is used. The former detects whether the tweet is emotional or not, and the tweets that have emotions, are driven to the input of the latter which classifies them as positive or negative. And in this way, graph-based hashtag level analysis of the tweets, which have emotions, is applied in order to achieve higher precision.

2.2 Cloud Computing Preliminaries

The Apache Spark is an open source framework designed specifically for cluster computing with the programming language Scala[1]. It is made so as to support general purpose distributed applications, based (generally) on the processing of large volume data, with a high degree of efficiency and speed. Regarding speed, Spark extends the popular model MapReduce in order to support more types of processing, such as interactive questions and flow data processing.

One of the main features of Spark is the ability to run calculations in memory. The Spark can accelerate an application to Hadoop cluster up to one hundred times with the memory usage, and ten times using only the disc. For ease of use, Spark offers simple APIs in Python, Java, Scala and SQL as well as rich embedded libraries. It can also be easily used with other big data tools like run on Hadoop cluster while having access to all Hadoop data. Finally, with regard to sophisticated analysis techniques, Spark supports SQL queries, data flow processing, and complex analytics such as machine learning algorithms and out-of-the-box. It also gives the users the ability to combine all these in a single program. The elements forming the Spark ecosystem are:

Spark Core is the core of the Spark and contains its basic functions. It includes the necessary elements for scheduling, memory management, fault repair, interaction with the system storage, and others.

Spark SQL provides SQL queries support as well as the neighboring language of SQL created by Apache Hive, called Hive Query Language-HiveQL. Apart from providing the SQL interface for the Spark, the Spark SQL enables developers to introduce SQL queries in Spark.

Spark Streaming is an element of the Spark that allows data flow processing in real time.

MLlib is a library of machine learning functions. The MLlib provides various machine learning algorithms, including binary classification, regression and collaborative filtering.

GraphX is a library that was added to the Spark 0.9 (February 2014) and it provides an API for manipulating graphs and performing calculations in parallel graphs. An example is the process of a graph representing users friends in a social network.

3 SENTIMENT CLASSIFICATION ON TWITTER

As aforementioned, in [23], a different approach for emotion analysis using a graph consisting of hashtags, is proposed. This model is called Hashtag Graph Model and examines the polarity of graph nodes. Initially, sentiment analysis on tweet level and in following on hashtag level is utilized; below these two steps are presented.

3.1 Tweet-Level Sentiment Classification

To categorize tweets, we have used a two-phase SVM classifier in order to estimate each sentiment deriving from them. In the first phase, a tweet is categorized as neutral or subjective while in the second phase, a subjective tweet is further categorized as positive or negative. Both SVMs use the same features, e.g. unigram words, punctuation and emoticons, as well as the sentiment lexicon.

3.2 Hashtag-Level Sentiment Classification

Given a graph model, the aim is to solve the assignment inference problem by proposing efficient algorithms. To this point, three approaches have been introduced, namely Loopy Belief Propagation (*LBP*), Relaxation Labeling (*RL*) and Iterative Classification Algorithm (*ICA*).

Hashtags can be categorized either as positive or negative. Suppose a set of hashtags $H = \{h_1, h_2, ..., h_m\}$, where each one is associated with a set of tweets $T = \{t_1, t_2, ..., t_n\}$. The aim is to create a set y, which contains the polarity of each hashtag throughout H. We consider that hashtags, which are in a specific tweet, are more likely to have the same polarity. This drives us to introduce the graph for the analysis to come. That is, a graph HG with hashtags is considered, entitled Hashtag Graph Model, as presented in

[1] http://www.scala-lang.org/

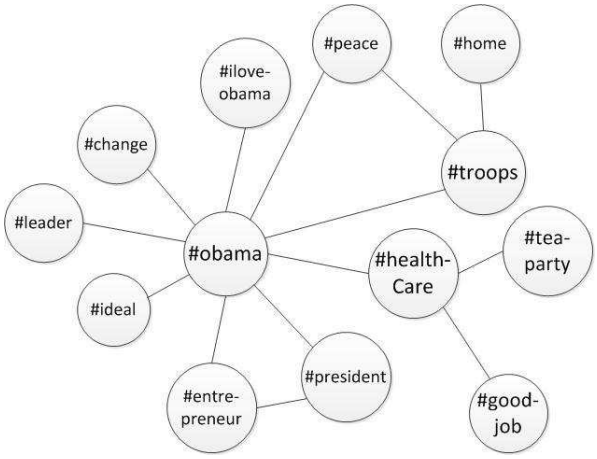

Figure 1: An example of Hashtag Graph Model [23]

1. The graph is defined as $HG = \{H, E\}$, where H is a set of hashtags and E a set of edges where each edge connects two hashtags that coexist in a tweet.

Our primary goal is to categorize each hashtag as positive or negative. So, we therefore use the following equation:

$$\log(Pr(y|HG)) = \sum_{h_i \epsilon H} \log(\phi_i(y_i|h_i)) + \sum_{(h_j,h_k)\epsilon E} \log(\psi_{j,k}(y_j, y_k|h_j, h_k)) - \log(Z)$$

wherein the first summation represents the tweet coefficient while the second the hashtag coefficient. Moreover, the Z is a normalization factor. From the above equation, we allow adjacent hashtags to affect the result of classification.

The final result derives from the maximization of the following function

$$\hat{y} = argmax_y log(Pr(y|HG)) \quad (1)$$

The three algorithms are presented below and the corresponding pseudocodes are also introduced in following.

3.2.1 Loopy Belief Propagation. *LBP* is an iterative algorithm and so, it tries to classify each node in a graph through belief message passing. It was originally proposed to work for tree−like networks as a Bayes likelihood-ratio updating rule. Although it does not guarantee convergence to a fixed point after any number of iterations, *LBP* shows surprisingly good performance in practice. In fact, the propagation process tries to reach the stationary points of the Bethe approximation free energy for a factor graph.

Initially, the algorithm initializes all edges of the graph with one message for both labels (positive-negative). The next step is to refresh the messages with the multiplication of functions ϕ and ψ and the messages of neighboring nodes. Finally, y is calculated for each node.

3.2.2 Relaxation Labeling. *RL* is another algorithm for classification on the graph. Specifically, the value of $d_i j$ is used in order to detect the "significance" of the node j with respect to the node i. In addition, $r(y_i, y_j)$ is used for estimating the compatibility between the labels y_i and y_j. One can also consider the probability $b_i(y_i)$ of hashtag h_i for having the tag y_i.

Loopy Belief Propagation

Input : Hashtag Graph **HG**
Output : Sentiment label for each hashtag h
begin
for all $(h_i, h_j) \in E$ **do**
  for all $y \in \{pos,neg\}$ **do**
    $m_{i\to j}(y) \leftarrow 1$
    $m_{j\to i}(y) \leftarrow 1$
  end for
end for
repeat
  for all $h_i \in H$ **do**
    for all $h_j \in N(h_i)$ **do**
      for all $y_j \in \{pos, neg\}$ **do**
        $m_{i\to j}(y_j) \leftarrow a \sum_{y_i} \psi_{i,j}(y_i, y_j)\phi_i(y_i) \prod_{h_k \in N(h_i)\setminus h_j} m_{k\to i}(y_i)$
      end for
    end for
  end for
until all $m_{i\to j}(y_i)$ stop changing;
for all $h_i \in H$ **do**
  $\hat{y}_i \leftarrow \arg\max_{y\in\{pos,neg\}} a\phi_i(y) \prod_{h_j \in N(h_i)} m_{j\to i}(y)$
  return $\{\hat{y}_i\}$
end for

3.2.3 Iterative Classification Algorithm. Algorithm *ICA* starts with the categorization of each hashtag by using a probability function of the tweet that contains it. After each iteration, it calculates again the probability of each node with respect to the probability of its neighbor and estimates the emotion of the top-k hashtags; k is increased linearly at each iteration.

4 SPARK IMPLEMENTATION

In this Section, the basic parts of the proposed algorithms are outlined. It has to be stated that our work is the first where these algorithms are implemented in Scala programming language, which is the most appropriate one for the creation of graph model.

Twitter Stream: Initially, the algorithm takes as input a word (or even a set of words) from the command line and in following, it gets the tweets that contain that specific word. Our experiments are based on Twitter and used Twitter API to collect tweets; specifically we implemented our Twitter network using Twitter4j[2] library. One hurdle we had to overcome was that the Twitter API has a number of limitations as it does not provide access to a large number of tweets.

Tweet Classification-MLlib: To categorize tweets, three operating classifiers with the help of MLlib[3] are used. MLlib is a Spark library which supports machine learning algorithms. During the

[2]http://twitter4j.org/en/index.html
[3]http://spark.apache.org/mllib/

Relaxation Labeling

Input : Hashtag Graph **HG**
Output : Sentiment label for each hashtag h
begin
for all $h_i \in H$ **do**
  for all $y_i \in \{pos,neg\}$ **do**
    $b_i(y_i) \leftarrow \frac{\sum_{\tau \in T_i} Pr_{y_i}(\tau)}{\sum_{\tau \in T_i} \sum_y Pr_y(\tau)}$
  end for
end for
repeat
  for all $h_i \in H$ **do**
    for all $y_i \in \{pos, neg\}$ **do**
      $q_i(y_i) \leftarrow \sum_{h_j \in N(h_i)} d_{i,j} \Big[\sum_{y_j} r(y_i, y_j) b_j(y_j) \Big]$
      $a_i \leftarrow \sum_y b_i(y) \Big[1 + q_i(y)\Big]$
      $b_i(y_i) \leftarrow \frac{1}{a_i} b_i(y_i) \Big[1 + q_i(y_i)\Big]$
    end for
  end for
until all $b_i(y_i)$ stabilize;
for all $h_i \in H$ **do**
  $\hat{y}_i \leftarrow \underset{y \in \{pos,\, neg\}}{\arg\max}\ b_i(y)$
end for
return $\{\hat{y}_i\}$

Iterative Classification

Input : Hashtag Graph **HG**
Output : Sentiment label for each hashtag h
begin
for all $h_i \in H$ **do**
  $y_i \leftarrow \arg\max_{y \in \{neg,pos\}} \phi(y|h_i)$
end for
for $t = 1 \rightarrow M$ **do**
  for all $h_i \in H$ **do**
    compute $p_i(y_i|\mathbf{(HG,y)})$
    Store $p_i \leftarrow \max_y p_i(y|\mathbf{HG,y})$
    Store $y_i \leftarrow \arg\max_y p_i(y|\mathbf{HG,y})$
  end for
  $k \leftarrow |H| \frac{t}{M}$
  Update hashtag labels with top-k p_i
end for
return $\{y_i\}$

training stage of the classifier, we have used a data set which consists of $1,600,000$ categorized tweets[4]. We processed the data with Spark−sql and we have used the Databricks[5] library in order to further edit the corresponding files; final target being sentiment prediction.

Hashtag Graph Model-GraphX: The creation of the graph is considered as an important step of our framework. In order to achieve this, we need three mutable lists, mutable meaning that the content of the list can be altered. In our case though, the creation of the Resilient Distributed Dataset (*RDD*) does not allow such alternation. Lists are for *TCID* links, edges and the number of hashtags. Some operations need to be run on order to bring the hashtag in desired form required by GraphX[6]. The disadvantage that GraphX does not support the import of additional nodes on a specific graph is surpassed due to the fact that the algorithm is in real time and as new tweets arrive, the new hashtags are added to the list and the graph is created from the beginning.

5 RESULTS AND EVALUATION

We have verified the results presented in [23] in terms of Accuracy, Precision, Recall and F1 evaluation metrics for the Apache Spark implementation. More specifically, the dataset used is the same as in [23]; 15.000 pre-classified tweets from a set of 600.000 ones all of which were categorized as positive, negative and neutral. The classifier used is the two-stage SVM Tweet-Level, whereas K-Fold Cross-Validation ($K = 5$ Fold) is performed.

Apart from verifying the results, we have also compared the SVM with Naive Bayes, Logistic Regression as well as Decision Trees classifiers. The results are presented in Table 1 Accuracy-wise where both achieve almost similar performance.

Table 1: Performance of Tweet-Level Classifiers

Classifier	Accuracy
2-phase SVM	84.13%
Naive Bayes	81.75%
Logistic Regression	72.45%
Decision Trees	76.23%

Since our motivation stems from the fact that we are interested in identifying a high number of correct classified tweets in all classes of each classifier, Table 1 verifies our goal, meaning that the accuracy of the four classifiers remains high. The same high performance in terms of precision and F1 shows the combination of subjectivity and polarity classifiers as shown in [23]. We can observe that SVM outperforms the other three algorithms achieving higher accuracy in a range from about 3% to 12%.

Specifically, as for the recall metric in [23], expressing the fraction of tweets in each class which is correctly classified, the results show that the subjectivity classifier finds difficulty in classifying correctly the tweets of subjective class. On the contrary, the polarity classifier achieves approximately $2x$ better performance in terms of recall metric.

6 CONCLUSIONS

In our work, we have developed four classifiers in Spark, namely SVM, Naive Bayes, Logistic Regression and Decision Trees. The classifiers were trained with the use of a large dataset of 1.600.000 pre-classified tweets where tweets were categorized as positive, negative or neutral. The achieved accuracy of classifiers proves

[4] http://www.sentiment140.com/
[5] https://databricks.com/
[6] http://spark.apache.org/graphx/

the classification efficiency and stability of results. All the classification algorithms are implemented in Apache Spark cloud framework using the Apache Spark's Machine Learning library, entitled MLlib.

As future work, we plan to further investigate the effect of different data features such as bigrams, trigrams, unigrams or POS tags as introduced in our previous work [1]. Moreover, collecting data from other sources, such as the posts from Facebook and Instagram or YouTube comments or even reviews from Foursquare, is a further investigated area to be examined. Furthermore, we aim at experimenting with different clusters and evaluate Spark's performance in regards to time and scalability as presented in our previous works in Spark environment [9, 21].

ACKNOWLEDGEMENT

This work was funded by General Secretariat for Research and Technology (GSRT) and Hellenic Foundation for Research and Innovation (HFRI) and supported by University of Patras.

REFERENCES

[1] A. Baltas, A. Kanavos, and A. Tsakalidis. 2016. An Apache Spark Implementation for Sentiment Analysis on Twitter Data. In *International Workshop on Algorithmic Aspects of Cloud Computing (ALGOCLOUD)*. 15–25.

[2] L. Barbosa and J. Feng. 2010. Robust Sentiment Detection on Twitter from Biased and Noisy Data. In *International Conference on Computational Linguistics: Posters (COLING)*. 36–44.

[3] A. Bermingham and A.F. Smeaton. 2010. Classifying Sentiment in Microblogs: is Brevity an Advantage?. In *ACM Conference on Information and Knowledge Management (CIKM)*. 1833–1836.

[4] D. Davidov, O. Tsur, and A. Rappoport. 2010. Enhanced Sentiment Learning Using Twitter Hashtags and Smileys. In *International Conference on Computational Linguistics: Posters (COLING)*. 241–249.

[5] D. Gayo-Avello, P. Metaxas, and E. Mustafaraj. 2011. Limits of Electoral Predictions Using Twitter. In *International Conference on Weblogs and Social Media (ICWSM)*.

[6] A. Go, R. Bhayani, and L. Huang. 2009. Twitter Sentiment Classification using Distant Supervision. *CS224N Project Report, Stanford* (2009), 1–6.

[7] M. Hu and B. Liu. 2004. Mining and Summarizing Customer Reviews. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 168–177.

[8] L. Jiang, M. Yu, M. Zhou, X. Liu, and T. Zhao. 2011. Target-dependent Twitter Sentiment Classification. In *Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 151–160.

[9] A. Kanavos, N. Nodarakis, S. Sioutas, A. Tsakalidis, D. Tsolis, and G. Tzimas. 2017. Large Scale Implementations for Twitter Sentiment Classification. *Algorithms* 10, 1 (2017), 33.

[10] A. Kanavos, I. Perikos, I. Hatzilygeroudis, and A. Tsakalidis. 2016. Integrating User's Emotional Behavior for Community Detection in Social Networks. In *International Conference on Web Information Systems and Technologies (WEBIST)*. 355–362.

[11] A. Kanavos, I. Perikos, I. Hatzilygeroudis, and A. Tsakalidis. 2018. Emotional Community Detection in Social Networks. *Computers & Electrical Engineering* 65 (2018), 449–460.

[12] A. Kanavos, I. Perikos, P. Vikatos, I. Hatzilygeroudis, C. Makris, and A. Tsakalidis. 2014. Conversation Emotional Modeling in Social Networks. In *IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*. 478–484.

[13] V.A. Kharde and S. Sonawane. 2016. Sentiment Analysis of Twitter Data: A Survey of Techniques. *International Journal of Computer Applications* 139, 11 (2016).

[14] A. Kumar and T. Sebastian. 2012. Sentiment Analysis on Twitter. *IJCSI International Journal of Computer Science Issues* 9, 3 (2012), 372–378.

[15] B. Liu. 2012. *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers.

[16] B. O'Connor, R. Balasubramanyan, B.R. Routledge, and N.A. Smith. 2010. From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. In *International Conference on Weblogs and Social Media (ICWSM)*. 122–129.

[17] A. Pak and P. Paroubek. 2010. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In *International Conference on Language Resources and Evaluation (LREC)*. 1320–1326.

[18] B. Pang and L. Lee. 2008. Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval* 2, 1-2 (2008), 1–135.

[19] B. Pang, L. Lee, and V. Shivakumar. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. In *ACL Conference on Empirical methods in Natural Language Processing*. 79–86.

[20] V. Sahayak, S. Vijaya, and P. Apashabi. 2015. Sentiment Analysis on Twitter Data. *Innovative Research in Advanced Engineering (IJIRAE)* 2 (2015).

[21] S. Sioutas, P. Mylonas, A. Panaretos, P. Gerolymatos, D. Vogiatzis, E. Karavaras, T. Spitieris, and A. Kanavos. 2016. Survey of Machine Learning Algorithms on Spark Over DHT-based Structures. In *International Workshop on Algorithmic Aspects of Cloud Computing (ALGOCLOUD)*. 146–156.

[22] M. Thelwall, K. Buckley, and G. Paltoglou. 2012. Sentiment Strength Detection for the Social Web. *International Journal of the American Society for Information Science and Technology (JASIST)* 63, 1 (2012), 163–173.

[23] X. Wang, F. Wei, X. Liu, M. Zhou, and M. Zhang. 2011. Topic Sentiment Analysis in Twitter: A Graph-based Hashtag Sentiment Classification Approach. In *ACM International Conference on Information and Knowledge Management (CIKM)*. 1031–1040.

[24] J. Wiebe, T. Wilson, and C. Cardie. 2005. Annotating Expressions of Opinions and Emotions in Language. *Language Resources and Evaluation* 39, 2-3 (2005), 165–210.

[25] W. Zhang, C. Yu, and W. Meng. 2007. Opinion Retrieval from Blogs. In *ACM Conference on Conference on Information and Knowledge Management (CIKM)*. 831–840.

[26] L. Zhuang, F. Jing, and X.Y. Zhu. 2006. Movie Review Mining and Summarization. In *ACM International Conference on Information and Knowledge Management (CIKM)*. 43–50.