Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 132 (2018) 956–964

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Intelligence and Data Science (ICCIDS 2018)

Real-Time Sentiment Analysis of Twitter Streaming data for Stock Prediction

Sushree Das[a], Ranjan Kumar Behera[a], Mukesh kumar[b], Santanu Kumar Rath[a]

[a] Department of Computer Science and Engg., National Institute of Technology, Rourkela, Odisha, India, 769008

[b] ABB GISPL, Bangalore-560048

Abstract

In this study, an attempt has been made for making financial decisions such as stock market prediction, to predict the potential prices of a company's stock and to serve the need of this, Twitter data [1 2] has been considered for scoring the impression that is carried for a particular firm. Streaming data proves to be a perennial source of data analysis collected in real-time. Streaming data basically deals with the continuous flow of data which carries information from sources like websites, mobile phone applications, server logs, social websites, trading floors, etc. The major characteristics of such data being its accessibility and availability, help in proper analysis and prediction of user behavior in a ceaseless manner. The classifying model made out of historical data can be relentlessly honed to give even more accurate results since its outcome is always compared to the next tick of the clock. Spark streaming has been considered for the processing of humongous data and data ingestion tools like Twitter API and Apache Flume have been further implemented for analysis.

Keywords: Spark Streaming, Sentiment Analysis, Recurrent Neural Network, Apache Flume

1. Introduction

As the technology advances, there has been a remarkable growth in the size of the data produced on a daily basis. Hence, the need of managing such humungous data on a real-time basis has cropped up. This paved the motivation for considering a data processing architecture for massive quantites of real-time data. In this study, a case on prediction of stock prices has been taken into consideration. The major requirements for a good real-time data processing architecture are such that it should be fault-tolerant, extensible and scalable [3]. After witnessing the use cases implemented by companies like Yahoo or Netflix, "Lambda Architecture" [4] was considered to be attempted for such analysis. The need for analysis based on huge amount of data has laid new foundations for research in areas such as Big Data analysis projects which resulted in a robust model that could scale up to handle with optimum fault tolerance [5 6]. In this rattling world, information flow has become quite easy. This includes the dissemination

Corresponding author. Tel.: +9-194-395-92352
E-mail address: sushreedas008@gmail.com

10.1016/j.procs.2018.05.111

of di erent market rates through financial websites or views of the users via social websites [7], both of which can influence the market rates through leaps and bounds. Hence, predicting the rates through these information makes it more meaningful since the continuous flow of data reshapes the model prepared from historical data, thus making it even more accurate with each iteration.

Analysis of financial data like, stock prediction predominantly relies on the idea of e cient market hypothesis [8] which implies that the current rise or fall of the stock prices of a company mostly depends on the impression of it on the public mind [9]. Bad reviews generally decrease the popularity of the markets reputation and a good review brings a better impact [?]. A proof to such hypothesis is the corporal example of Tesla in which a positive review regarding a new motor launch of the company increased its stock market value to a multiple fold [10]. A lot of researches have already proved that this data has helped analysis to decide upon various fields like epidemic spread analysis [11], box-o ce collection [12], polling results prediction [13] etc. The basic idea behind this methodology is to detect the mood of the customers or users and then predict on the parameter in subject.

There are many ways of ingesting information by considering data from various services, for example, Flume, Zero Message Queue(ZMQ), Twitter API, Kinesis, and Kafka [14 15]. TCP attachments may also be processed by the utilization of complex calculations that are communicated using abnormal state capacities. Once the information is processed, the knowledge may be pushed to databases, live dashboards, and document frameworks like Amazon's Simple Storage Service(S3) [16] and Hadoop Distributed File System(HDFS) [17]. Apache Spark has been used for the analysis of streaming data. Apache Spark is a well-known open-source platform that processes large scale data and analyses them through a component called "MLlib" i.e., Spark's distributed machine learning library [18 19]. The reason as to why it has been chosen in this study is that Apache Spark is a programming model that involves Discretized Streams in the form of Resilient Distributed Dataset(s) [20 21]. RDDs ensure the fault handling and the latency of the system. MLlib's tight reconciliation with Spark brings about a few advantages. To begin with, since Spark is planned in light of iterative calculation, it empowers the advancement of productive executions of extensive machine learning calculations since they are iterative in nature. Enhancements in low-level parts of Spark frequently converts them into performance gains in MLlib, with no immediate changes to the library itself. Second, Spark's open source group has prompted fast development and selection of MLlib including commitments from more than 140 individuals. Third, MLlib is one of few eminent libraries based on best of Spark. This study aims to predict the movement of opening stock price for a particular company on a particular day with the help of the above discussed underlying architecture that would help in predicting the data on the flow.

Section 2 explains the first phase of the model which is Data Ingestion. This study has considered Twitter data for analyzing the sentiment of the users for a particular company using various machine learning techniques. Ingestion of data has been done using the developer tool available on Twitter i.e. Twitter Streaming API. The collected data is then dumped into HDFS through Apache Flume. Subsequent sections provide a brief idea on how the ingested data is analysed using tools such as MLlib. The results section thus indicates that the proposed architecture for the prediction purpose is the one which is scalable, fault tolerant and robust.

2. Methodology

2.1. Framework for Streaming Analysis

This section highlights on prediction of stock market, based on a certain data processing architecture known as Lambda Architecture. It is an open-source, fault tolerant model that can be scaled up based on requirement with a promise of lower latency and robustness. This architecture predominantly includes three layers namely, the batch layer, speed layer and the serving layer.

The ingested data is pushed into the batch and the speed layer simultaneously. Batch layer stores the history data which is immutable and constantly grows with every second. It also helps in building precomputed models based on those history data. Now, this data is not that frequently precomputed, rather done in several time-frames which have a longer duration. The second layer which is the speed layer helps in decreasing the latency of the overall model by working only with the recent data. And finally serving layer helps in indexing the historical data so that information can be easily retrieved through indexes itself.

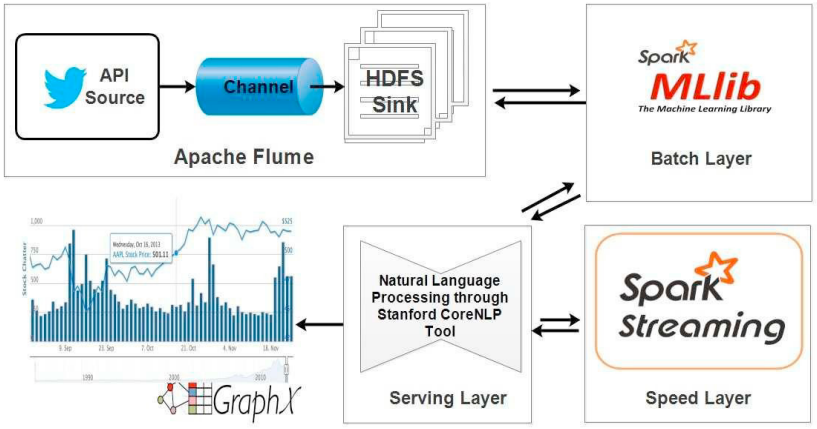


Fig. 1: Architecture for Streaming Data Analysis

To actualize this model, Twitter API has been considered for ingesting data. The ingested data is then stored in HDFS through the help of Apache Flume, wherein Twitter API acts as a source and HDFS acts as a sink respectively. For working on the historical data, Apache Spark has been considered for implementation which takes input JavaScript Object Notation(JSON) formatted data. Streaming data is ingested for every second in the form of tweets per second and is worked upon through Spark streaming. The overall model for analyzing the sentiment is done through Stanford Core NLP tool [22] which helps in correctly classifying sentiment of each tweet into three notable classes such as Positive, Negative and Neutral. Various tools that are being used in the study to realize the underlying architecture have been presented in Fig. 1.

2.2. Data Ingestion Through Twitter API

For analyzing financial data, which consists of huge amount of streaming data in the form of event data (tweets in this case), Apache Flume has been used as the tool for ingestion mechanism. Since the rate of incoming data varies with every tick of the clock, Flume mediates between data producer and the centralized stores through the memory channel making it a reliable source for data transferring. The pivotal daemon process called the Flume Agent receives events from sources such as Twitter API and forwards it to the sink (HDFS in this case). Interceptors, a component of the Flume Agent, inspects the incoming events which are then transferred through the channel. It ensures fault handling by having two transactions per event. Preliminary naming of the components for the realization of the ingestion part has been shown in Fig. 2.

```
TwitterAgent.sources = Twitter
TwitterAgent.channels = MemChannel
TwitterAgent.sinks = HDFS
```

Fig. 2: Components used for Data Ingestion

After listing out the components for the ingestion process, individual property description is made for each of the components. Enumeration of one snippet is shown in Fig. 3.

```
TwitterAgent.sources.Twitter.type = Twitter (type name)
TwitterAgent.sources.Twitter.consumerKey = GeneratedFromApp
TwitterAgent.sources.Twitter.consumerSecret = GeneratedFromApp
TwitterAgent.sources.Twitter.accessToken = GeneratedFromApp
TwitterAgent.sources.Twitter.accessTokenSecret = GeneratedFromApp
```

Fig. 3: Description of Component Feature

3. Time-Series Analysis with Twitter sentiments

Time-series information is pervasive crosswise over numerous areas, including financial markets, medicinal services, meteorology, seismology, and space science. Information visualization (InfoVis) and visual analytics (VA) methods are mostly used to break down such time-series datasets [23]. They help us in better use of the human capacity to increase understanding from perceptions, including distinguishing the fundamental patterns, irregularities, and con-nections, through an intelligent and undirected inquiry. This predicted data have been incorporated into time-series portrayals before [24]. Hu et al. have named this approach a visual expectation i.e., the demonstration of outwardly foreseeing a time-series variable by watching the forecasts from a computational model, parallel to time-series rep-resentations [10]. Conventional methodologies may not completely bolster visual investigation of future patterns in complex multivariate datasets, for example, securities exchanges, climate, and human services information, since it is devoid of considering the inter-variable relationships. In order to aid such analysis, microblogging sites provide a tremendous source of information in the real-world.

As, in this paper, it is focussing on the financial data analysis, the Twitter Application Programming Interface(Twitter API) which provides a streaming API, has been considered that renders data continuously. Each individual data col-lected represents the state or mood of the user regarding a particular theme. This can be fetched through an elemental HTTP authentication and a twitter account. After retrieving all of the data for every tick of data, analysis of the senti-ments related to each tweet is initiated and thus the mood which has a direct implication on the status of the stock is being predicted. Sentiment analysis is basically a classication problem in which the input data having a positive or a negative sentiment are classified. Several models are prepared based on di erent learning algorithms that are applied to the training data. After preparing such a model, the streaming data are collected through the Streaming API.

3.1. Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) are sought-after models that have demonstrated extraordinary promise in numerous NLP tasks [25]. RNN is often considered for training the model in a recursive manner considering the sequential pattern in historical data. Fig. 4 shows how Recurrent Neural Network is being unfolded into a full network.

The definitionof the variables used in RNN are discussed as below:

i_t is the input at time step t.

h_t is the hidden state at time step t. It represents the memory of the network. h_t is calculated based on the hidden state prior to the current state and the input at the current step:

$$h_t = f(N\ i_t + M\ h_{t\ 1}) \quad (1)$$

The function f usually is a nonlinear function such as tanh or ReLU. $h_{t\ 1}$ which is required to calculate the first hidden state, and typically initialized to all zeros.

o_t is the output at step t.

Key things to be noted here is that the parameters such as M or N are the same and only the input parame-ters change over each iteration. The training of RNNs is similar to other neural networks. At each time step t

(also referred to frame), this network receives the inputs as well as its own output from the previous time step, o_t. This mechanism is generally termed as unrolling the network through time. Following equation symbolises the output for a single instance:

$$o_t = f(i_t \; N + o_{t\;1} \; M + b) \tag{2}$$

where b is the bias of the network.

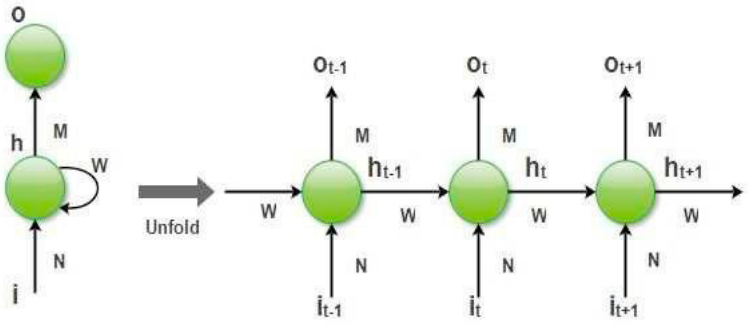


Fig. 4: Recurrent Neural Network (RNN)

For this experiment, the first step helps to create word vectors each with a dimensionality of 300. Furthermore, an embedding matrix is developed for each word so as to realise the impact or sentiment it has for each of the tweets it has been used in. So for every tweet, such embedded matrices are developed and each word is given as input at each time step. This helps to evaluate the overall sentiment for every tweet that has been captured in the ingestion process. The training set that is being used contains twitter data i.e. tweets that involve the ticker keyword (for example #apple or #aapl in this case). The dataset procured contains 56000 tweets which range over the span of thirteen months of twitter data starting from May, 2016 till June, 2017. The training set contains an equal set of correctly classified positive and negative reviews about the firm currently under scrutiny. This gives the average length of words too that later empowers to calculate the polarity of each tweet.

4. Experimental Analysis

Twitter sentiment has been performed by using RNN components in Standford Core NLP. Standford Core NLP is a standard natural language software used for extracting various form of sentiments from large set of text. It allows the user to model NLP classifier in scalable manner that acts as reference architecture.

4.1. Batch Processing Layer

The sentiment analysis of history data was done via one of the Spark's components called the MLlib which exploits the iterative computation done through Spark. Naive Bayes classifier has been considered which is an RDD based API for this experiment. They can be used by importing the packages as shown in Fig. 5. These classes ensure certain functionalities which involve loading the training set and mapping them into classes after the model is trained.

4.2. Stream Processing and Serving Layer

For obtaining the sentiments on a real-time basis, Stanfords CoreNLP API has been used. This involves the usage of RNN which has been discussed in the aforementioned sections. It is fundamentally based on JVM annotation pipeline framework that provides most of the popular natural language processing steps. Three datasets have been considered, each of which containS data filtered based on the tags of each company. In this experiment,

prediction of the movement of stock prices for three major companies has been taken into consideration, namely, Google, Microsoft and Apple Inc.

```
import org.apache.spark.mllib.classification.{NaiveBayes, NaiveBayesModel}
import org.apache.spark.mllib.util.MLUtils
```

Fig. 5: Importing Naive Bayes Component of Spark-Mlib

```
val sentiment = RNNCoreAnnotations.getPredictedClass(tree)
```

Fig. 6: Twitter Sentiment Prediction

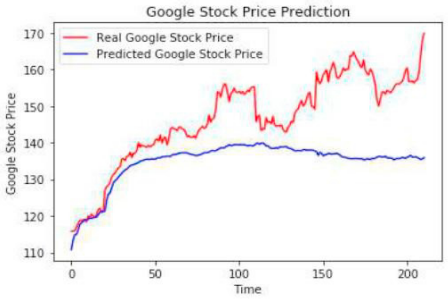
For this, at each ticker, tweets that had keywords like MSFT,GOOG,AAPL, MICROSOFT,GOOGLE or APPLE have been filtered. Then they were segregated and collected in three di erent datasets. Later, batch data from online sources like Yahoo Finance were collected for that particular timeframe and company. The main class that is to be defined for such a classification is SentimentAnalysisUtils wherein the definitions of the properties such as “annotators” and “tokenize, ssplit, pos, lemma, parse, sentiment” have been set. After importing all the necessary packages from edu.stanford.nlp, the code snippet shown in Fig. 6 has been used to predict the sentiment for a particular tweet at any time stamp. This enables to calculate the weighted sentiment for each tweet. Final classification of the sentiment can be done as per method shown in Table. 1.

Table 1: Categorical Classification for the Sentiments

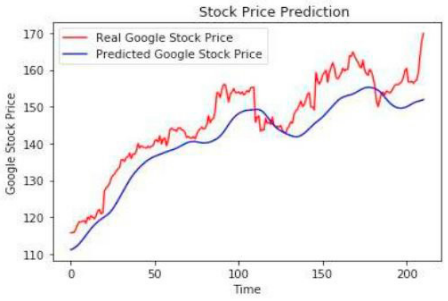
Sl.no	Condition	Prediction
1	S 0:0	NOT UNDERSTOOD
2	0:0 < S 1:0	VERY NEGATIVE
3	1:0 < S 2:0	NEGATIVE
4	2:0 < S 3:0	NEUTRAL
5	3:0 < S 4:0	POSITIVE
6	4:0 < S 5:0	VERY POSITIVE
7	S 5:0	NOT UNDERSTOOD

5. Result and Discussion

Actual Stock price of Google, Microsoft and Apple have been collected using Yahoo Finance website which is treated as ground truth for performance measure. The predicted stock price for the testing data is compared with the actual price. The dataset procured contains 5,60,000 tweets which range over the span of thirteen years of twitter data starting from May, 2005 till Jun, 2017 and was stored in HDFS. The training set contains an equal set of correctly classified as positive and negative reviews about the firm under consideration. Twitter data collected over last 200 days are taken as testing dataset. Prediction graph for each of the history dataset grouped based on company names was plotted along with the plot that shows the weighted polarity of the tweets, made on that particular day for that particular company. Fig 7a and Fig. 7b show the correlation between actual and predicted stock price of Google for 150 and 300 epoch respectively. The correlation graph for Microsoft in 150 and 300 Epoch is shown in Fig. 8a and Fig. 8b respectively.

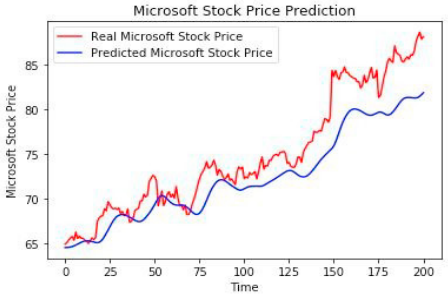


(a) Prediction for 150 Epoch

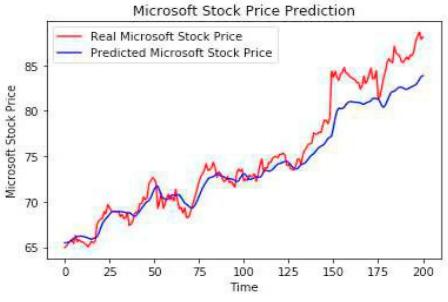


(b) Prediction for 300 Epoch

Fig. 7: Correlation between predicted and actual stock price for Google

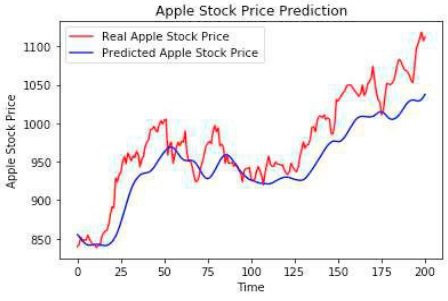


(a) Prediction for 150 Epoch



(b) Prediction for 300 Epoch

Fig. 8: Correlation between predicted and actual stock price for Microsoft



(a) Prediction for 150 Epoch

(b) Prediction for 300 Epoch

Fig. 9: Correlation between predicted and actual stock price for Apple

6. Threat to Validity

The proposed model ingests data from social networking sites and predicts the direction of movement of the opening price based on the sentiments of the tweets that were shared regarding it. The amount of speculation about a particular company may not be the same always thus making it di cult in certain timelines to predict regarding the direction of flow.

8

Due to unavailability of data or minimal availibility of data at certain timelines, the performance of the proposed model may not always be competetive.

Spark's MLlib is being used in the model for prediction of opening prices for the day which is still under development. This limits the user to select only the algorithms which are supported by MLlib.

7. Conclusion and Future Work

Predicting future values of stock prices is an interesting research area, commonly connected to the analysis of public mood. Given that more and more personal opinions are made available online, various studies indicate that these kinds of analyses can be automated and can produce useful results. The study indicates that sentiment analysis of public mood derived from Twitter feeds can be used to eventually forecast movements of individual stock prices. Furthermore, the methodology was adapted to a stream-based setting using the incremental active learning approach, which provides the algorithm with the ability to choose new training data from a data stream for hand-labeling. Stream based active learning for sentiment analysis of microblogging messages in the financial domain may contribute both to sentiment analysis and the active learning research area. Moreover, feasibility study through batch processing has also been performed with this experiment by taking the aid of RNNs Long Short-Term Memory(LSTM). This helps to further dig upon the streaming of online stock data available in various nancial websites which shall provide a better model to analyse and predict the future stock prices for a particular company. Thus, a hybrid model based on analysis of the sentiments and the current stock trend in the rise or fall of prices can be introduced which will improve its reliability as well as trust-worthiness of the prediction.

In future, inclusion of eclectic machine learning algorithms can be done for the prediction of stock data such as Deep learning models. Other methods of data ingestion such as data ingestion through Apache Flume or NodeJS can also be implemented and their performance and accuracy may be compared for the same.

References

1. Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. LREc, 10, 2010.
2. Xue Zhang, Hauke Fuehres, and Peter A Gloor. Predicting stock market indicators through twitter i hope it is not as bad as i fear. Procedia-Social and Behavioral Sciences, 26:55–62, 2011.
3. Michael Stonebraker, Uur C¸etintemel, and Stan Zdonik. The 8 requirements of real-time stream processing. ACM Sigmod Record, 34(4):42– 47, 2005.
4. Michael Hausenblas and Nathan Bijnens. Lambda architecture. URL: http://lambda-architecture. net/. Luettu, 6:2014, 2015.
5. Philip Russom et al. Big data analytics. TDWI best practices report, fourth quarter, 19:19–40, 2011.
6. Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. Future generation computer systems, 29(7):1645–1660, 2013.
7. Gema Bello-Orgaz, Jason J Jung, and David Camacho. Social big data: Recent achievements and new challenges. Information Fusion, 28:45–59, 2016.
8. B. G. Malkiel. The e cient market hypothesis and its critics. Journal of economic perspectives, 17:59–82, 2003.
9. De Rijke M. Mishne, G. Capturing global mood levels using blog posts. AAAI spring symposium: computational approaches to analyzing weblogs, 6:145–152, 2006.
10. Weiming Hu, Tieniu Tan, Liang Wang, and Steve Maybank. A survey on visual surveillance of object motion and behaviors. IEEE Transac-tions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 34(3):334–352, 2004.
11. Eiji Aramaki, Sachiko Maskawa, and Mizuki Morita. Twitter catches the flu: detecting influenza epidemics using twitter. In Proceedings of the conference on empirical methods in natural language processing, pages 1568–1576. Association for Computational Linguistics, 2011.
12. Carl Sabottke, Octavian Suciu, and Tudor Dumitras. Vulnerability disclosure in the age of social media: Exploiting twitter for predicting real-world exploits. In USENIX Security Symposium, pages 1041–1056, 2015.
13. Aibek Makazhanov, Davood Rafiei, and Muhammad Waqar. Predicting political preference of twitter users. Social Network Analysis and Mining, 4(1):193, 2014.
14. Nishant Garg. Apache Kafka. Packt Publishing Ltd, 2013.
15. Pieter Hintjens. ZeroMQ: messaging for many applications. " O'Reilly Media, Inc.", 2013.
16. Mayur R Palankar, Adriana Iamnitchi, Matei Ripeanu, and Simson Garfinkel. Amazon s3 for science grids: a viable solution? In Proceedings of the 2008 international workshop on Data-aware distributed computing, pages 55–64. ACM, 2008.
17. Ranjan Kumar Behera, Abhishek Sai Sukla, Sambit Mahapatra, Santanu Ku Rath, Bibhudatta Sahoo, and Swapan Bhattacharya. Map-reduce based link prediction for large scale social network. 2017.
18. Abdul Gha ar Shoro and Tariq Rahim Soomro. Big data analysis: Apache spark perspective. Global Journal of Computer Science and Technology, 15(1), 2015.

19. Ranjan Kumar Behera, SK Rath, and Monalisa Jena. Spanning tree based community detection using min-max modularity. Procedia Computer Science, 93:1070–1076, 2016.
20. Matei Zaharia, M Chowdhury, T Das, A Dave, J Ma, M McCauley, M Franklin, S Shenker, and I Stoica. Resilient distributed datasets. A Fault Tolerant Abstraction for In-Memory Cluster Computing, nd http://www. cs. berkeley. edu/˜ matei/talks/2012/nsdi_rdds. pdf, accessed April, 2014.
21. Matei Zaharia, Tathagata Das, Haoyuan Li, Scott Shenker, and Ion Stoica. Discretized streams: An e cient and fault-tolerant model for stream processing on large clusters. HotCloud, 12:10–10, 2012.
22. Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. The stanford corenlp natural language processing toolkit. In Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations, pages 55–60, 2014.
23. Leishi Zhang, Andreas Sto el, Michael Behrisch, Sebastian Mittelstadt, Tobias Schreck, Rene´ Pompl, Stefan Weber, Holger Last, and Daniel Keim. Visual analytics for the big data eraa comparative review of state-of-the-art commercial systems. In Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on, pages 173–182. IEEE, 2012.
24. Thomas Niederkrotenthaler and Gernot Sonneck. Assessing the impact of media guidelines for reporting on suicides in austria: interrupted time series analysis. Australian & New Zealand Journal of Psychiatry, 41(5):419–428, 2007.
25. Nikolaos Nodarakis, Spyros Sioutas, Athanasios K Tsakalidis, and Giannis Tzimas. Large scale sentiment analysis on twitter with spark. In EDBT/ICDT Workshops, pages 1–8, 2016.