An Improved Social Media Analysis on Three Layers: A Real Time Enhanced Recommendation System

Mohamed Amine TALHAOUI, Hicham AIT EL BOUR, Reda MOULOUKI, Saida NKIRI, Mohamed AZOUAZI
Laboratoire Mathématiques Informatique et Traitement de l'Information MITI
Hassan II University, Faculty of Sciences Ben m'Sik Casablanca, Morocco

***Abstract*—The Internet can be considered as an open field for expression regarding products, politics, ideas, and people. Those expressive interactions generate a large amount of data pinned per users and groups. In that scope, Big data along with various technologies, such as social media, cloud computing, and machine learning can be used as a toolbox to make sense of the data and give the opportunity to generate efficient analysis and studies of the individuals and crowds regarding market orientation, politics, and industry. The recommendation system for this acts as the pillars of technology, in the field of sentiment analysis and predictive analysis to make sense of user's data. However, this complex operation comes at the price of this. To each analysis, there is a personalized architecture and tool. In this paper, a novel design of a recommender system is provided powered by sentiment analysis and predictive models applied onto an example of data flow from the social media Twitter.**

Keywords—Twitter; machine learning; sentiment; Lambda; recommendation; Big data; opinion mining

I. INTRODUCTION

A significant amount of available data on Internet represent a rich source of knowledge extraction, in particular, the social media like Twitter or Facebook. How this data processing is performed enables data scientists to deliver relevant and exploitable results for economic, social, industrial, government policies or business purposes.

In this paper, the work is focused on providing for a company valuable information about its products. A proposition on how to extract data from Twitter based on keywords and store it in Hadoop Ecosystem [1] to carry out a sentiment analysis. This processing allows us to categorize tweets as positive, negative or neutral using scoring methods. The proposed system provides as well as a relevant way to detect the qualities and defects of the specific product using Word Clouds. Machine Learning is also a crucial aspect of this work; it enables us to predict Satisfied customer, Customer not satisfied, prospect and suspect for a distinct product.

For that several techniques are used:

A. Machine Learning

The purpose of machine learning [2] is to induce a system into learning from the past or present and apply that knowledge to make predictions or decisions regarding unknown future events. In the most general terms, a supervised machine learning task is composed of three stages: build the model, assess and tune the model, and next put the model into production.

After an analysis of some ML supervised algorithms such as:

- Naive Bayes (NB)
- Support Vector Machine (SVM)
- Neural Networks (NN)

A choice to use NB is because it has been shown to perform surprisingly well with minimal amounts of training data that most other classifiers, would find significantly insufficient.

Naïve Bayes is a classification [3] algorithm that relies on strong assumptions of covariance independence in the Bayes theorem. The Naïve Bayes classifier assumes independence between dependent predictive variables and a Gaussian distribution of the numerical predictors with mean and standard deviation calculated from the training dataset. Naïve Bayes models are commonly used as an alternative to decision trees for classification problems. When constructing a Naïve Bayes classifier, each row of the training dataset containing at least one NA will be ignored completely. If the test data set has missing values, these predictors are omitted in the probability calculation during prediction.

B. Opinion Mining

Sentiment analysis [4] or Opinion mining can be described as the computational study of people's opinions, judgments, attitudes, and emotions toward entities and their aspects. The entities usually relate to products, services, organizations, individuals, events, etc. and the aspects are attributes or components of the entities. With the growth of social media (i.e., reviews, forum discussions, and blogs) on the Web, individuals and organizations are increasingly using the opinions in these media for decision-making.

Sentiment detection [5], as usual, is a classic problem of text classification. Unlike other text classification tasks, the goal is not to identify topics, entities, or authors of text but to rate the expressed sentiment typically as positive, negative, or neutral. Most approaches used for sentiment detection have also been used for other text classification tasks and usually involve techniques from machine learning, computational linguistics, and statistics. Typically, different approaches from these fields are used for sentiment detection. Linguistic considerations [6] range from tokenizing the to-be- classified texts to other syntactic analyses. Statistical considerations typically involve frequencies of tokens or phrases, e.g., the occurrence of many "positive" words in a text, or similar

statistics. The respective features then usually are coupled with machine learning algorithms to classify the sentiment of arbitrary texts.

Due to the diverse applications [7] in mining and retrieval, and since Twitter is one of the most abundant sources of opinion, a lot of different approaches to sentiment detection in tweets have been proposed. Various methods use different feature sets ranging from standard word polarity expressions or unigram features also applied in general sentiment detection, to the usage of emoticons and uppercases, word lengthening, phonetic features, multi-lingual machine translation, or word embedding. The task usually is to detect the sentiment expressed in a tweet as a whole. But it can also be to identify the sentiment in a tweet concerning a given target concept shown in a query. The difference is that a negative tweet might not say anything about the target concept and must thus be considered neutral concerning the target concept.

In this section, a detailed study of the different technologies used to elaborate the architecture followed by a section to go through the components and specify the role of each one then the third section to present the proposed architecture as well as the algorithms modeled to achieve it. Section four will establish the fundamental view of the results obtained and the last section to conclude and discuss the next perspectives.

II. Technological Spectrum

A. Ease of Use Big Data and Components

1) Hadoop distributed file system (HDFS)

A file system designed [8] to store massive volumes of data across various commodity hardware configurations called nodes.

2) MapReduce

An engine [9] for data processing. A MapReduce job consists of two components, a map phase, which takes raw data and arranges it into a paradigm of key/value pairs, and a reduce stage which processes data in parallel.

3) Apache Pig

A tool/platform [10] that can analyze large datasets and represent them as data streams.

4) Flume

Flume, developed initially by Cloudera, is today a project [11] of the Apache software foundation.

Flume is a product that can inject large volumes of data into Hadoop in real time. By design, close to that of an HDFS cluster, Flume is:

- Reliable: in case of failure of one of its components, Flume can continue to feed HDFS.
- Evolutive: Flume performance can be increased by adding scaling outs.
- Extensible: by default, Flume can ingest data from various sources (local files, HDFS files, system logs, stdout ...) and, if necessary, additional connectors can be developed.

Flume is composed of agents. Each agent has a source (source), a destination (sink) and a channel:

A source can be a data source (firewall, mail server, web server ...) or another agent.

A destination may be another agent or an HDFS file.

A channel is a path followed by data between a source and a destination: a channel can write its data to RAM or disk, depending on the user's needs regarding performance and reliability, the process is described in Fig. 1.

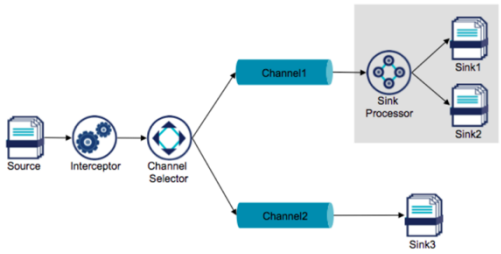

Fig. 1. Flume data process.

Flume offers different levels of reliability:

- Support: In the event of a node failure, all data being received or sent by this node is lost; it is the mode that offers the least guarantees and generates the least overhead.
- Store on failure: In this mode, a source node expects to receive an acknowledgment from the destination node; if not, the data to be transferred is saved on the hard disk until the destination node returns "life sign" or is replaced by another node. This mode does not protect against silent or cascade failures.
- End-to-end: This method ensures that data supported by a source node will reach its final destination, provided that the source node does not experience a crash immediately after data support. This is the mode that offers the most guarantees and generates the most overhead.

5) Mahout

One of the significant well-known tools [12] for ML. It is known for having a wide selection of robust algorithms, math environment named Samsara, which carries linear algebra, statistical operations, and data structures. The goal of the Mahout-Samsara project is to help users build their own distributed algorithms, rather than just a library of already-written implementations. They still propose a comprehensive suite of algorithms for MapReduce.

The algorithms included in Mahout, focus primarily on classification, clustering, and collaborative filtering, and have been shown to scale as much as the size of the data increases. Additional tools include topic modeling, dimensionality reduction, text vectorization, similarity measures, a math library, and more. One of Mahout's most usually cited assets is its extensibility as shown in Fig. 2 as a comparative study of the different ML frameworks, and many have achieved positive results by building o of the baseline algorithms.

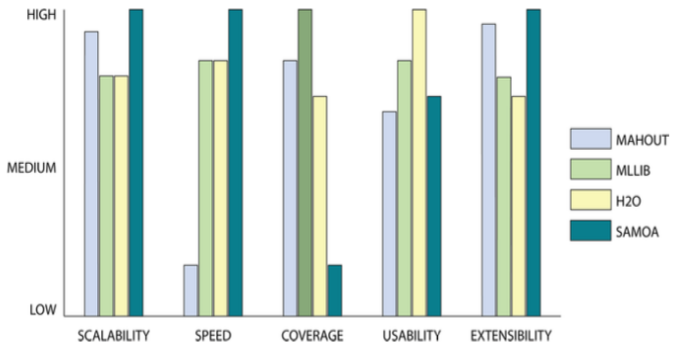

Fig. 2. Mahout vs the others (comparative).

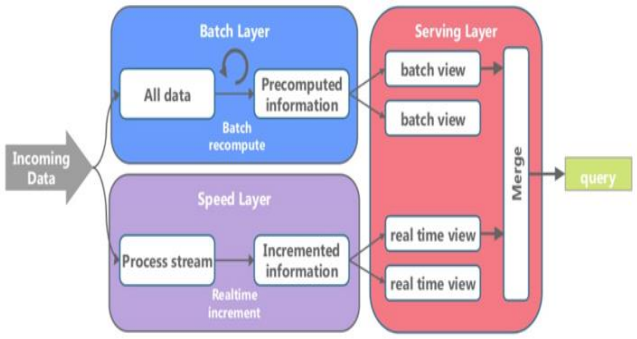

Fig. 3. Lambda architecture layers.

III. Approach

Data analysis is essential for planning and creating recommendation systems or decision-making systems to optimize the underlying infrastructure. This involves not only the processing of data online, looking for specific events but also the historical data sources that may be needed to find data profiles influencing decisions.

Researchers have often merged techniques with other tools to develop field-related solutions. In particular [13], the Twitter API had been extended to third researchers to deploy their analysis of flow data from Twitter to improve business practices. However, unique solutions that allow multiple users in different environments to write and use optimized data processing applications are still needed. Nevertheless, there is a need for tailored solutions for online and batch data processing that maintain non-functional attributes such as network costs and complexities.

Presented as a software design model, the Lambda architecture [14] unifies real-time and batch processing in a single framework. It was founded to be a hybrid system. The model is suitable for applications where there are delays in the collection and availability of data in dashboards, which requires the validity of data for on-line processing as it happens. The model also allows batch processing for older data sets to find patterns of behavior according to user needs.

Some of the critical requirements in the construction of this architecture include:

- Fault tolerance against hardware breakdowns and human mistakes.
- Support for a multiple diversity of use cases that involve low latency queries and updates.
- Linear scalability, which means that the launch of additional machines will solve the problem without degrading the performance of resources.
- Scalability so that the system is manageable and can quickly adapt to new features.

From a high-level viewpoint, the figure below shows the underlying architecture of how the lambda architecture works.

Three layers are treated as detailed in Fig. 3:

1) Batch layer

Batch deduct for large quantities of data sets. It provides management of the Master Dataset; an immutable and exclusive raw data set, but also includes pre-computation of arbitrary query functions, called batch views.

2) Serving layer

Real-Time calculation (Speed Time) to minimize latency by performing calculations in real time as the data arrives. This layer indexes the batch views so that they can be queried in Ad-Hoc with low latency.

3) Speed layer

Responses to requests, interfacing, query and providing the results of calculations. This layer accepts all requests that are subject to low latency requirements. Using fast and incremental algorithms, Speed Layer processes only recent data.

Each of these layers can be obtained using various large data technologies.

To technically reflect the aspect of the Lambda Architecture in the solution provided in Fig. 4 this equivalent Lamda design with the respective big data component to each layer.

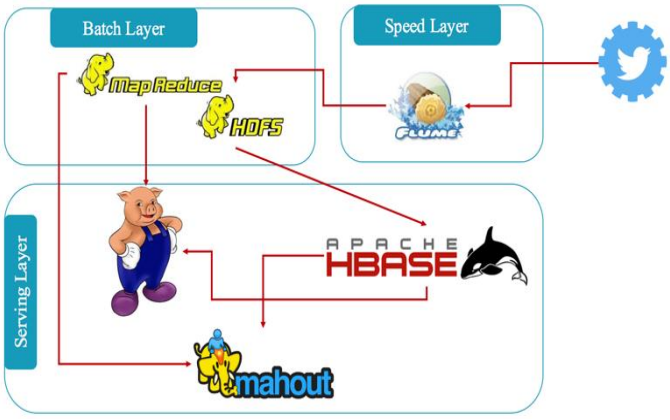

Fig. 4. Equivalent Lambda architecture.

IV. Solution

As a solution, this architecture describing the process of the several steps from capture to data analysis in Fig. 5:

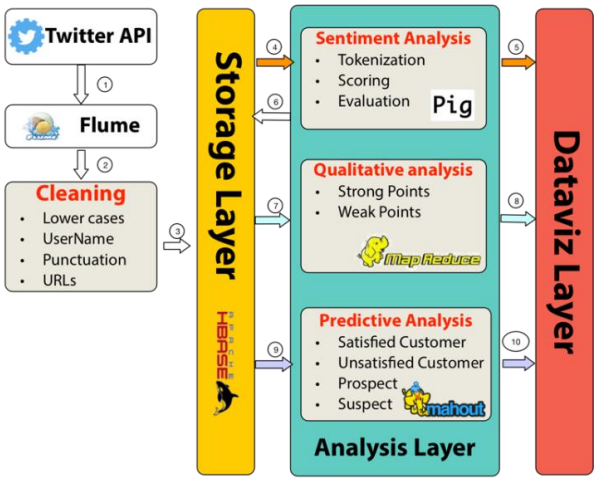

Fig. 5. Solution architecture process.

A. Data Capture

The development of data mining applications requires a covering of issues related to data access, mainly when application policies result in a lack of data to analyze. In other words, the data is not always open.

The exploitation of social networks is a little less worrying compared to other corporate environments as most social media platforms offer well-designed language agnostic APIs that provide access to the data needed.

The availability of this data obviously depends on how users share their data and how they allow us to access it. For example, Facebook users can decide the level of privacy of information about their public profile and the details that can be displayed only to their friends. Profile information such as birthday, current location and work history (and many others) can all be reported individually as private or public. Similarly, when trying to access this data through the Facebook API, users who sign up to the application can grant us access only to a limited subset of the requested data.

On the other hand, the Twitter API is open and allows access to all user data. Using Flume, by retrieving data from various services and transport them to centralized stores (HDFS and HBase) [15]. More precisely, extracting data from the Twitter service and store it in HDFS using Apache Flume as illustrated in the following Fig. 6.

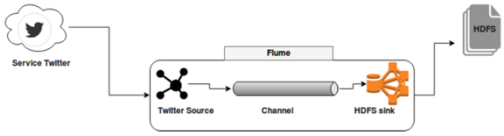

Fig. 6. Twitter data capture.

To get Twitter tweets, it is necessary to create a Twitter application, then start the services of Hadoop and finally configure Apache Flume.

B. Data Pre-processing

Data pre-processing involves eliminating unwanted fields and special characters so that the feeling analysis is as reliable as possible. This is called noise elimination or cleaning. To do this, the following adjustments will be satisfied:

- LowerCase: To convert tweets to lowercase.
- URLs: To remove all URLs via a match or a regular replacement expression with a generic word URL.
- @username: To remove "@username" via regex matching or replace it with a generic word like AT-USER.
- Additional Punctuation and White Spaces: Punctuation at the beginning and end of the tweets should be removed. We also replace several white spaces with unique white space.
- Eliminate unnecessary fields: As previously stated, the tweet contains information and fields that will not provide any benefit to

C. Storage

As previously explained, storage must be at HBase level. Now all tweets are cleaned and ready to be stored. To get started, firstly by preparing the database and configure it, create region servers, master backups, and tables. The files are stored in the tables of HBase, and the tables can be managed using the 'hbase shell' commands.

D. Data Analysis

1) Opinion Mining

The opinion mining will be done with the Apache Pig tool. This one will provide us the classification of the tweets according to the feelings; positive if the polarity of the total words constituting the tweet is positive, neutral if it is zero and negative elsewhere. The data (tweets) are stored in Hbase. To perform sentiment analysis, using the Hbase table where the tweets are stored. The first thing to do with a tweet is what is called Tokenization or splitting.

This operation consists of dividing a string (which contains a group of words) into a row and returns a result holding the output of the split operation. Here is an illustration of the output of this treatment.

- First, starting with loading tweets in Pig, and the next step is the extraction of the id and the text of the tweet from the complete tweet.
- Working with the function tokenize of Pig which refers to the words needed to work with.
- Now since having the tweets or rather the words, the treatment sentiment analysis can start.
- This processing is done as follows, and we classify the words treated from -5 to 5 using the AFFIN dictionary [16] (is a dictionary that contains 2500 words ranged from -5 to 5: from 0 to -5 it is negative, 0 neutral, from 0 to 5 this is positive).
- Loading the AFFIN dictionary.
- The next step is to link each token of the previous step by its ranking word which is in the AFFIN dictionary.
- Extracting that id, text, and score.
- The next step is the calculation of the score of the whole tweet, starting with the grouping of the words of each tweet by the id of the latter:

- Calculating the average of all the tweet, And since having the average, the filter can be applied according to user's needs.

Performing the rating on all the words of each tweet as follow:

Algorithm Algorithm for input and sentiment analysis

Input: KeyWords

Output: (id,sentiment)

Extraction :

1: Cleaning

Tokenization :

2: Token(tweet)

Loading :

3: Load(AFFIN Dictionary)

Rating :

4: Rate(word ,dictionary)

Group Rating :

5: GroupRate(id ,word)

LOOP Process

6: **for each** GroupRate **do**
7: AVGrate tweetRating
8: **if** (tweetRating > 0) **then**
9 print(id ,positive)
10: **else**
11: **if** (tweetRating < 0) **then**
12 print(id ,negative)
13 **else**
14 print(id ,neutral)
15: **end if**
16: **end if**
17: **end for**

2) Qualifying Analysis

The qualification service is a service that allows visualizing the different keywords with varying sizes according to their frequencies in the set of tweets. Indeed, it is also a MapReduce program which looks for the occurrence of a keyword in the tweets and calculates its frequency of occurrence, then gives each word a value.

This value represents the proportion of size with which it will be written or displayed. The processing is done on all tweets, resulting in a Wordcloud. As sentiment analysis processing, it is an R program that will be able to perform a Word Cloud based on the output of the MapReduce program that is stored in a file in HDFS. Indeed, R contains a library called 'wordcloud', the latter as its name indicates it is responsible for the creation of WordCloud.

3) Predictive Analysis

This service is dedicated to the prediction of user profiles such as Satisfied Customer, Unsatisfied Customer, Prospect, Suspect.

Mahout is the right tool for this system, since it implements the Naive Bayes algorithm, which was chosen as the prediction algorithm during the comparative study, to define the classification mechanism, because the algorithm is of the supervised type, we will start with a set of observations of many variables, until we can assign a new observation to a particular category.

Using this set, the classifier determines the probability that a user belongs to one of the declared categories for each word. To calculate the probability that a user belongs to one of the categories, it multiplies the individual probability of each of its elements in that category. The category with the highest probability is the category that the user is most likely to belong to.

V. Experimental Evaluation

All the experiments were performed using a 2,5 GHz Intel Core i7 processor and 16GB of RAM running on a Cloudera virtual machine.

A. Opinion Mining

From this analysis, we will get all tweets, both positive and negative. To classify them, the positive tweets are those with a rating between 1 and 5. Obviously, the negative tweets will have a score between -5 and -1. The rest is considered neutral.

In Fig. 7, we can see the tweet text field and its score. We managed to classify the tweets according to the emotion in the text field.

```
((argan and grapeseed what a lovely combination I bet it smell wonderful), 3.
5)
((wonderful natural hair oil for damaged hair damaged hair growth argan gomed
iderm), -0.5)
((give away give away give away giveaway argan arganoil  I hate oil),-4.0)
((fx11 argan based mascara to help your eyelashes flourish! pop in today and
try),1.5)
((try the organic natural argan face cream from aromabguk order from us moroc
co ),0.5 )
((love argan oil? remember thait only made in morocco if icomes from another
country its probably nothe real thing), 2.0)
```

Fig. 7. Sentiment analysis

Now all that remains is to present them on a graph. It is a MapReduce program that will do this; it counts all tweets with a positive score and those with a negative rating, then passes the result to a program written in R and finally has an understandable diagram. In this work, here is the result obtained in Fig. 8.

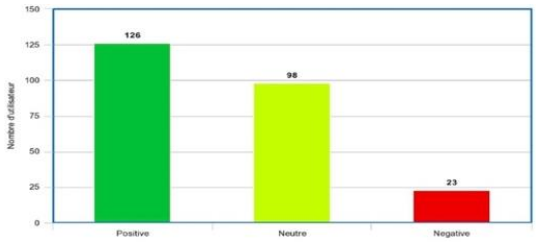

Fig. 8. Scoring Algorithm results

B. Qualifying Analysis

To go further, we were able to isolate positive tweets from negative tweets and then perform the same treatment on each of them. This will allow us to detect problems with the product just by looking at the negative WordCloud. On the other hand, it will help us to know the strong points of the product through WordCloud positive. Here is the result in Fig. 9.

Fig. 9. WordCloud result.

C. Predictive Analysis

Practically speaking, we started by preparing a Training Set. The most tedious task was to manually sort all tweets by category. We have chosen to classify a hundred tweets retrieved via the keyword (number of elements recommended for the correct functioning of the algorithm), the result of this sorting is stored in HDFS, in fact, it is a file that contains the following fields: User id that published the tweet, tweet feeling (positive, negative) obtained from the analysis of the previously processed feeling, another field that indicates if the user is already a customer of the company or not, and finally, to transform his data into a Training Set, we add the last field that represents the profile of the user (satisfied customer, unsatisfied customer, prospect, and suspect). The classifier determines for each keyword, the probability that a user belongs to one of the categories considered (satisfied customer, unsatisfied customer, suspect, and prospect) which allows us to deduce the profile of the person who published this tweet.

Table I below shows an extract of the data from the Training set:

TABLE I. TRAINING SET DATA

Ids	Sentiment	Client	Profil
id1	Positive	Yes	Satisfied
Id2	Negative	Yes	Non Satisfied
Id3	Negative	No	Suspect
Id4	Positive	No	Prospect
…	…	…	…

Finally, we use Mahout to transform this Training Set to a vector, the tweets-vectors directory of the HDFS contain sequences on which the Naive Bayes algorithm will be based for its calculations as described in Fig. 10.

```
$ mahout testnb -i train-vectors -m model -l labelindex -ow -o tweets-testing -c
[...]
Summary
-------------------------------------------------------
Correctly Classified Instances          :        135       97.8260%
Incorrectly Classified Instances        :          3        2.1739%
Total Classified Instances              :        138

=======================================================
Confusion Matrix
-------------------------------------------------------
a       b       c       d       <--Classified as
33      0       0       0       |  33          a     = clientSatisfait
0       18      0       1       |  19          b     = clientNonSatisfait
0       0       48      0       |  48          c     = prospect
1       1       0       36      |  38          d     = suspect
```

Fig. 10. Final classified results.

VI. CONCLUSION

In this paper, we provided a new architecture for a real-time recommendation system based on a social media case study: Twitter.

This architecture provides a new vision of massive data processing and opens the door to new research horizons. Analyzing the full scenario for social data processing.

The recommendation system applications have been designed and implemented in such a way that they complement each other, to become involved in a functional structure for Big Data analysis. The system has demonstrated its ability to perform a quantitative (and qualitative) analysis of the user's behavior to better understand their needs.

As a new perspective, we will concentrate the research on the user behavior field and loyalty by implementing the aspects of gamification for an improved recommendation.

REFERENCES

[1] R. Mukherjee, "A Comparative Review Of Data Warehousing ETL Tools With New Trends And Industry Insight," 2017 IEEE 7th Int. Adv. Comput. Conf., pp. 944–949, 2017.

[2] T. P. Pavlic and S. C. Pratt, Handbook of Human Computation. 2013.

[3] J. Chin, V. Callaghan, and I. Lam, "Understanding and personalising smart city services using machine learning, The Internet-of-Things and Big Data," 2017 IEEE 26th Int. Symp. Ind. Electron., pp. 2050–2055, 2017.

[4] H. Blockeel and K. U. Leuven, Encyclopedia of Machine Learning and Data Mining. 2017.

[5] N. Schrape, Gamification and Governmentality. 2014.

[6] D. Hutchison, LNCS 8462 - Persuasive Technology 2014. 2014.

[7] E. Fredericks and D. R. Schneider, Supporting Real Time Decision-Making, vol. 13. Boston, MA: Springer US, 2011.

[8] Y. Mao and W. Min, "Storage and Accessing Small Files Based on HDFS," in Proceedings of International Conference on Computer Science and Information Technology, 2014, pp. 565–573.

[9] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Proc. 6th Symp. Oper. Syst. Des. Implement., pp. 137–149, 2004.

[10] S. Wadkar and M. Siddalingaiah, "Data Processing Using Pig," in Pro Apache Hadoop, Berkeley, CA: Apress, 2014, pp. 241–269.

[11] D. Vohra, "Apache Flume," in Practical Hadoop Ecosystem, Berkeley, CA: Apress, 2016, pp. 287–300.

[12] S. Rallapalli, R. R. Gondkar, and G. V. Madhava Rao, "Cloud Based K-Means Clustering Running as a MapReduce Job for Big Data Healthcare Analytics Using Apache Mahout," 2016, pp. 127–135.

[13] J. Littman, D. Chudnov, D. Kerchner, C. Peterson, Y. Tan, R. Trent, R. Vij, and L. Wrubel, "API-based social media collecting as a form of web archiving," Int. J. Digit. Libr., Dec. 2016.

[14] J. Kroß, A. Brunnert, C. Prehofer, T. A. Runkler, and H. Krcmar, "Stream Processing on Demand for Lambda Architectures," 2015, pp. 243–257.

[15] N. Maheswari and M. Sivagami, "Large-Scale Data Analytics Tools: Apache Hive, Pig, and HBase," in Data Science and Big Data Computing, Cham: Springer International Publishing, 2016, pp. 191–220.

[16] S. Paul and S. Deshmukh, "Domain Independent Joint Sentiment And Topic Detection Using AFINN And Other Lexicons," IOSR J. Comput. Eng. Ver. I, vol. 17, no. 4, pp. 2278–661, 2015.