Knowl Inf Syst (2019) 59:413–436
https://doi.org/10.1007/s10115-018-1212-z

REGULAR PAPER

Sentiment analysis using semantic similarity and Hadoop MapReduce

Youness Madani[1] · **Mohammed Erritali**[1] · **Jamaa Bengourram**[1]

Received: 18 December 2017 / Revised: 4 April 2018 / Accepted: 6 May 2018 /
Published online: 18 May 2018

Abstract Sentiment analysis or opinion mining is a domain that analyses people's opinions, sentiments, evaluations, attitudes, and emotions from a written language; it had become a very active area of scientific research in recent years, especially with the development of social networks like Facebook and Twitter. In this paper we propose two new approaches to classify the tweets (look for the feeling expressed in the tweet), the first according to three classes : negative, positive or neutral, and the second according to two classes : negative or positive. Our first method consists in calculating the semantic similarity between the tweet to classify and three documents where each document represents a class (contains the words that represent a class); after the calculation of the similarity, the tweet takes the class of the document that has the greatest value of the semantic similarity with it. And the second method consists in calculating the semantic similarity between each word of the tweet to classify and the words "positive" and "negative" by proposing a new formula. We decide to do the analysis in a parallel and distributed way, using the Hadoop framework with the Hadoop distributed file system (HDFS) and the programming model MapReduce to solve the problem of the calculation time of the analysis if the dataset of the tweets is very large. The aim of our work is to combine between several domains, the information retrieval, semantic similarity, opinion mining or sentiment analysis and big data.

Keywords Opinion mining · Sentiment analysis · Semantic similarity · WordNet · Big data · Hadoop

Youness Madani, Mohammed Erritali and Jamaa Bengourram have contributed equally to this work.

✉ Youness Madani
younesmadani9@gmail.com

Mohammed Erritali
m.erritali@usms.ma

Jamaa Bengourram
bengoram@yahoo.fr

[1] Department of Computer Sciences, Faculty of Sciences and Technics,
Sultan Moulay Slimane University, Beni Mellal, Morocco

1 Introduction

With the explosion of internet and social networks, it emerged the need to analyse millions of posts, tweets or opinions in order to know what users think; opinion mining or sentiment analysis automatically analyses all these textual data and highlights the different opinions expressed on a specific topic such as a brand, a new item or a product. Sentiment analysis is considered as machine learning because it allows classifying the people's feelings towards any topic or contextual polarity of any text (containing people opinions) by using natural language processing techniques.

Today social networks become an important environment where people express their feelings, emotions and attitudes about products, brands, films, political phenomena, etc., in a free way which makes it easier to find the opinions of people on a product, for example, before buying it.

Twitter is the most used social network in opinion mining field, Twitter is a social network of microblogging with over 302 million active users per month and about 500 million tweets per day, its users can publish messages called tweets, and the number of characters of each tweet is limited to 140 characters and may include text, URLs, other user mentions (@) and hashtag (#) metadata to messages.

The analysis of feelings of tweets has become a very active field of scientific research in recent years especially with the increase of the users in twitter and also the number of tweets published daily to express a reviews or an opinion on something, for that many scientists seek to propose new approaches to give solutions to the problems related to this area of research.

Social data are growing rapidly in size, variety and complexity, so it is the problem of the large number of tweets published every day for example about a product, and therefore, it is the problem of time required to have the result of the classification of tweets because the classification process has many tasks to do before seeing the classification result (positive, negative, neutral).

To remedy this problem, we propose to work in a Big Data system by working with the Hadoop framework, with its distributed file system (Hadoop distributed file system (HDFS)) to distribute the storage of the tweets to classify and also the classification result, and its programming model Hadoop MapReduce to develop the classification.

In this work we propose new approaches to classify tweets according to three classes: positive, negative and neutral that is a subjectivity classification, or according to two classes positive, negative that is a polarity classification, in a parallel way by sharing the classification work between a set of machines (Hadoop cluster); our first approach consists in calculating the semantic similarity between each tweet to classify and three documents; each one contains words that represent a class, that is, a document for the positive class and one for the negative class and the other for the neutral class; after the semantic similarity calculation, the tweet takes the class of the document which has the high value of the semantic similarity with it. The second approach consists in proposing a new formula which consists of calculating the subtraction between the sum of semantic similarity of each word of the tweet and the word "positive" and the sum of the semantic similarity of each word of the tweet and the word "negative," after that if the value found is positive the tweet is positive, and if it is negative the tweet is negative.

Our method is therefore to combine several domains, such as semantic similarity with the use of WordNet, information retrieval systems, sentiment analysis or opinion mining and Big Data.

The rest of this paper is organized as follows: Sect. 2 presents the state of the art of the sentiment analysis, Sect. 3 gives a brief introduction of the notions of the information retrieval systems and semantic similarity and describe how we choose the semantic similarity approach, in Sect. 4 we going to present our two proposed methods and our research methodology, in Sect. 5 we describe how we parallelize our approaches using HDFS and MapReduce, and finally in Sect. 6 the conclusion with some perspectives is drawn.

2 Literature review

Scientific research for sentiment analysis knew a great development in recent years with the emergence of social networks and forums where users express their opinions, emotions and attitudes in a free way, which allows collecting a large volume of data and opinions on products and social phenomena in general; the analysis of the opinions of users knows many problems to find methods that are very effective and that give good results.

The usage of social networks in sentiment analysis grows explosively, and many researchers use them in different topics, for example for predicting an election results, extraction opinions about a product or a brand, or in an e-learning system as presented in [1], where authors proposed a new adaptive e-learning system, in which they analyse the personality of a learner to know his motivation (motivate, demotivate, neutral) using his twitter's profile by classifying the tweets that publish in his account in a specific period of the day, and based on his motivation and his profile, they give him courses adapted to his profile. Authors in this work use the AFINN Dictionary for the classification of the tweets.

Sentiment classification techniques can be roughly divided into machine learning approach, lexicon-based approach and hybrid approach [2]. The machine learning (ML) approach applies the famous ML algorithms (SVM, Naive Bayes, neural network, etc.) and uses linguistic features. The machine learning algorithms are typically used as classifiers for identifying sentiment or opinion and assigning a label (typically positive or negative) to a piece of text. The Lexicon-based approach relies on a sentiment lexicon, a collection of known and precompiled sentiment terms. It is divided into dictionary-based approach and corpus-based approach which use statistical or semantic methods to find sentiment polarity [3]. The hybrid approach combines both approaches, and it is very common with sentiment lexicons playing a key role in the majority of methods. Orestes et al. [4] present a hybrid approach to the sentiment analysis problem at the sentence level. This new method uses natural language processing (NLP) essential techniques, a sentiment lexicon enhanced with the assistance of SentiWordNet, and fuzzy sets to estimate the semantic orientation polarity and its intensity for sentences, which provides a foundation for computing with sentiments.

In the literature, we find many works that try to propose new approaches for improving the quality of the sentiment analysis methods. Authors in [5] show how the emoji characters on social networks can affect the text mining and sentiment analysis, for that, they decide to use the social network Twitter for collecting text data for some global positive and negative events to analyse the impact of emoji characters in sentiment analysis. They use the "New Year's Eve" as an event with positive feelings and "Istanbul Attack" as a negative event. As experimental results authors observed that using emoji in sentiment analysis helps improve overall sentiment scores and also the usage of them in sentiment analysis appeared to improve the expressivity and overall sentiment scores of the positive opinions relatively more than the negative opinions in our analysis.

Rezwanul Huq et al. [6] develop a functional classifier which can correctly and automatically classify the sentiment of an unknown tweet. They propose techniques to classify the sentiment label accurately for that they introduce two methods: one of the methods is known as sentiment classification algorithm (SCA) based on k-nearest neighbour (KNN), and the other one is based on support vector machine (SVM). Authors in this work classify tweets into two classes: positive and negative. Their key parameters to evaluate the performance of the classifications are accuracy, recall and precision on 1000 tweets, and the experiments show that sentiment classifier algorithm (SCA) performs better than SVM.

Pak and Paroubek [7] describe how to automatically collect a corpus for sentiment analysis and opinion mining, and authors build a sentiment classifier, that is able to determine positive, negative and neutral sentiments for a document. They use emoticons for collecting negative and positive sentiments and for collecting objective posts (no sentiments); they retrieved text messages from Twitter accounts of popular newspapers and magazines, such as "New York Times," "Washington Posts," for the classification; they used the multinomial Naive Bayes classifier, SVM and CRF; experimental results show that the Naive Bayes classifier gives the best results.

In [8] authors implement a two-step sentiment detection framework: the first step consists of distinguishing subjective tweets (sentimental tweets) from non-subjective tweets (objective tweets) that are a subjectivity detection, and the second step classifies the subjective tweets that express sentiments into positive and negative classes that is a polarity classification, and they used a variety of features like unigrams, bigrams, part-of-speech tags.

Agarwal et al. [9] developed a three-way model for classifying sentiment into three classes: positive, negative and neutral; they introduced POS-specific prior polarity features and the use of a tree kernel to obviate the need for tedious feature engineering; authors in this work present two classification methods: the first is a binary method for classifying sentiment into positive and negative classes, and the second is a three-way method for classifying sentiment into positive, negative and neutral classes; for the experimental results, they experiment with three types of models: unigram model, a feature-based model and a tree kernel-based model; experiment results show that a unigram model is indeed a hard baseline achieving over 20% over the chance baseline for both classification tasks. Their feature-based model that uses only 100 features achieves similar accuracy as the unigram model that uses over 10,000 features. Their tree kernel-based model outperforms both these models by a significant margin.

Before classifying a tweet, there is a step that is very important and that reduces the noise exists in the text, and it should help to improve the performance of the classifier and speed up the classification process; it is the text preprocessing, which is a set of treatments applied on the tweet like splitting, normalization, stemming, including removing URLs, replacing negation, reverting repeated letters, removing stop words, removing numbers and expanding acronyms; in the literature there are many works that study the effect of text preprocessing in sentiment analysis to know.

The work of ZHAO and GUI [10] UI in which authors try to demonstrate the role of text preprocessing methods to increase the quality of the sentiment classification in two types of classification tasks, for that they used six text preprocessing methods with two feature models and four classifiers based on five Twitter datasets. The experimental results show that the use of the text preprocessing methods of expanding acronyms and replacing negation improves the accuracy and F1 measure of the Twitter sentiment, and with the use of removing URLs, removing numbers or stop words methods the quality of classification was rarely changed.

Haddi et al. [11] explored the role of text preprocessing in sentiment analysis; they started by applying transformation on the data, which includes HTML tags cleanup, abbreviation expansion, stop words removal, negation handling and stemming, in which they use natural language processing techniques to perform them; the experimental results show that the accuracy of sentiment classification may be significantly improved using appropriate features and representation after text preprocessing.

Saif et al. [12] described how the process of removing stop words can improve the accuracy of sentiment analysis; they applied six different stopword identification methods to Twitter data from six different data sets and observe how removing stop words affect two well-known supervised sentiment classification methods. They assessed the impact of removing stop words by observing fluctuations in the level of data sparsity, the size of the classifier's feature space and its classification performance. Using precompiled lists of stop words negatively impacted the performance of Twitter sentiment classification approaches.

The authors in [13] studied the impact of text preprocessing methods on the classification of Twitter feelings, for that they use five methods of text preprocessing (the effects of URLs, negation, repeated letters, stemming and lemmatization) for remarking their effects. Experimental results show that the use of URLs features reservation, negation transformation, repeated letters and normalization rises the sentiment classification accuracy on the Stanford Twitter sentiment dataset while descends when stemming and lemmatization are applied. Moreover, authors get a better result by augmenting the original feature space with bigram and emotions features.

In [14] authors analyse the effect of negation in sentiment analysis, and they proposed a new approach to deal with negation in tweets to improve the classification accuracy, this approach is helpful for calculating the negation in sentiment analysis without the words not, no, n't, never, etc. This method produced a significant result for review classification by accuracy, precision and recall. The experimental results showed that if negation is ignored in a review the precision is 79%. With negation result improved as 84.87% precision. Their modified negation approach also improves the recall and accuracy as 84.81 and 91.8% than without negation classification.

On the other hand, in recent years, several authors use semantics in sentiment analysis to reduce problems related to natural language processing (NLP), and the experimental results show that the use of semantics in sentiment analysis improves the results of classification of the tweets. Authors in [15] proposed a new approach which adds the semantics for the classification as additional features into the training set for sentiment analysis; results show an average increase of F harmonic accuracy score for identifying both negative and positive sentiment of around 6.5 and 4.8% over the baselines of unigrams and part-of-speech features respectively. After that, they compare her approach against an approach based on sentiment-bearing topic analysis and find that semantic features improve recall and F score in negative sentiment classifying and also improve precision with lower recall and F score in positive sentiment classification.

In [16] the researchers proposed a novel approach that automatically captures patterns of words of similar contextual semantics and sentiment in tweets; their approach extracts patterns from the contextual semantic and sentiment similarities between words in a given tweet corpus; they evaluate their approach with tweet and entity-level sentiment analysis tasks by using the extracted semantic patterns as classification features in both tasks for that they use 9 Twitter datasets in their evaluation and compare the performance of their patterns against six state-of-the-art baselines. Results show that their approach outperforms all other similar approaches on all datasets by 2.19% at the tweet level and 7.5% at the entity level in average F measure.

Authors in [17] proposed a semantic approach to discover user attitudes and business insights from social media in Arabic, both standard and dialects. They also introduce the first version of their Arabic Sentiment Ontology (ASO) that contains different words that express feelings and how strongly these words express these feelings. They then show the usability of their approach in classifying different Twitter feeds on different topics.

As presented in this section, most existing approaches use the well-known machine learning algorithms for classifying tweets or sentences into classes. Our work consists in proposing new approaches for classifying tweets (positive, negative or neutral) without using machine learning algorithms. The idea is to take advantages of the semantic and the notions of the information retrieval system for proposing optimal methods that give good results if we compare them with those using the machine learning algorithms.

3 Information retrieval system and semantic similarity

As we said earlier, the aim of our work is to propose two methods to classify the tweets—the first method is to calculate the semantic similarity between the tweet to classify and three documents, where each document represents a class (positive, negative or neutral); our goal is to give the tweet the class of the document that has the great value of the similarity with it, so our proposal is like an information retrieval system where we want to find the documents that are relevant to a user's need expressed by a query. By comparing all this with our work, the tweet to classify will play the role of the query and the three documents that represent the three classes will play the role of the database of the documents that we seek.

3.1 Information retrieval system

The information retrieval system (IRS) is a system that tries to find a need of information of a user expressed by a query from a large volume of data that is mostly in the form of documents. In the information retrieval systems, we find a query and from it, we want to find the objects (documents) that are relevant to it, the way to evaluate a document if it is relevant or not to the user's need (query) is to calculate the similarity between the request and that document.

Before the calculation of the similarity, it is important to index all the documents and also the request that is to make them in a new presentation to facilitate their use, for example the vector representation [18], where each element of the vector represents the weight (frequency or the number of occurrences) of each term or concepts in the document or in the query. It is also important to make all the text preprocessing methods to facilitate the search of the relevant document for the query.

Figure 1 shows the important steps of an information retrieval system.

From this figure and as a comparison with our work, the query is the tweet to classify, and the database of documents is in our case the three opinion documents that represent the three classes (positive, negative or neutral); for the similarity calculation we use the semantic similarity using the semantic resource WordNet,[1] and one of the existing approach in the literature that calculates the semantic similarity.

3.2 Semantic similarity

Our two proposed methods for classifying tweets are both based on the calculation of semantic similarity; in this subsection, we present some approaches exist in the literature that calculates the semantic similarity.

[1] https://wordnet.princeton.edu/.

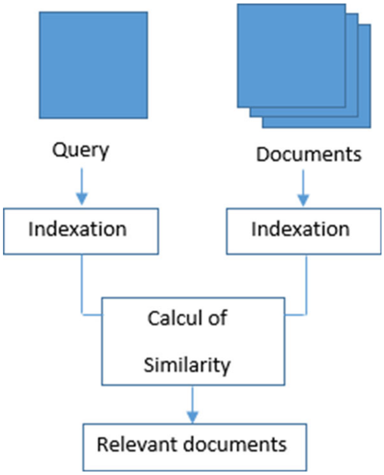


Fig. 1 Important steps of an information retrieval system

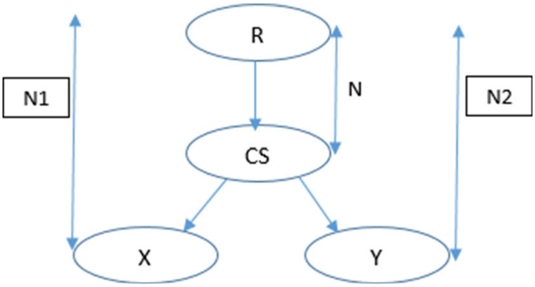


Fig. 2 Example of an ontology extract

The similarity measure of Wu and Palmer [19] is based on the following principle: given an ontology W formed by a set of nodes and a root node R (Fig. 2). X and Y are two elements of the ontology that we want to calculate their similarity. The principle of the similarity calculation is based on the distances ($N1$ and $N2$) which separate the nodes X and Y from the root node (R) and the distance between the subsuming concept (CS) or most specific concept (CPS) of X and Y from the node R.

The measurement of Wu and Palmer is defined by the following formula:

$$Sim_{\text{Wu \& Palmer}} = \frac{2 * N}{N_1 + N_2} \tag{1}$$

Resnik [20] has used the notion of semantic distance in the following way: two concepts are more similar if the value of the semantic distance between them is small. The similarity is defined in relation to the length of the paths that connect two concepts in the hierarchy. The similarity between two concepts c_1 and c_2 is:

$$Sim(c_1, c_2) = 2 * D - Len(c_1, c_2) \tag{2}$$

where D is the maximum length of possible paths that connect c_1 and c_2, and $Len(c_1, c_2)$ is the smallest path between c_1 and c_2.

Lin [21] has defined a different similarity measure that of Resnik by this formula:

$$Sim_{Lin} = \frac{2 * \log(P(\mathrm{CS}(c_1, c_2)))}{\log(P(c_1)) + \log(P(c_1))} \tag{3}$$

This measure uses a hybrid approach that combines two different sources of knowledge (Thesaurus, corpus).

Jiac [22] brought a new formula which consists in combining the entropy (Informational Contents) of the specific concept to those of the concepts which we seek the similarity. This approach is computed by the following formula:

$$Sim_{jiac}(c_1, c_2) = \frac{1}{distance(c_1, c_2)} \tag{4}$$

The distance between c_1 and c_2 is computed by the following formula:

$$distance(c_1, c_2) = E(c_1) + E(c_2) - (2 * E(\mathrm{CS}(c_1, c_2))) \tag{5}$$

In [23] authors proposed a new measure of semantic similarity, The basic idea of this measure is that if two concepts are linked together by a very short path and which “does not change the direction” then the two concepts are similar. The calculation of the similarity is based on the weight of the shortest path leading from one concept to another and the number of direction’s changes.

$$Sim_{Hirst-st}(c_1, c_2) = T - \mathrm{PCC} - K * D \tag{6}$$

where T and K are constants, PCC is the distance of the shortest path in the number of arcs, and D is the number of direction’s changes.

Another method to calculate the semantic similarity is that proposed by Leacock and Chodorow [24] which is based on the length of the shortest path between two synsets of WordNet. The authors limited their attention to “is-a” hierarchical links as well as the path length by the overall depth P of the taxonomy. The formula is defined by:

$$Sim_{lc} = -\log\left(\frac{cd(c_1, c_2)}{2 * M}\right) \tag{7}$$

where M is the length of the longest path that separates the root concept of the ontology from the concept more down. We denote by $cd(c_1, c_2)$ the length of the shortest path that separates c_1 from c_2.

3.3 Choice of the approach

The choice of the approach from those exists in the literature is a very important task in our work because it plays an important role when classifying tweets. To facilitate the choice of an approach than another (the optimal approach), we try to calculate the semantic similarity between two identical documents and choosing the approach or approaches that give good results. Table 1 illustrates the results obtained.

From this table we remark that the best approach is the Leacock and Chodorow approach which gives the greatest similarity with minimal execution time, and also the approach of Wu and Palmer gives good results.

So from these results we choose to work with the **Leacock and Chodorow** approach in the classification step.

Table 1 Similarity and execution time in ms with different approaches

Measurement Approaches	Similarity	Execution time (ms)
Leacock and Chodorow	0.14	1016
Wu and Palmer	0.13	1297
Resnik	0.04	1360
JiangConrath	0.04	1391
Lin	0.02	1344

4 Research methodology

In this section, we present the methodology of our work with the different steps for classifying tweets (extracting emotions and attitudes); our work consists of proposing two new approaches to classify the tweets, one according to three classes (positive, negative and neutral) and the other according to two classes (positive and negative). The classification is made semantically using WordNet and the **Leacock and Chodorow** approach, and in a parallel manner by parallelizing the classification between several machines using a Hadoop cluster with Hadoop distributed file system (HDFS) for storing the tweets to classify and the result of the classification, and MapReduce programming model for the development and parallelization of our work. Our two approaches are based on the English language.

Each proposed approach represents a type of classification of the sentiment analysis; the first approach that classifies tweets into three classes (positive, negative and neutral) is **subjectivity classification** that is to say, it classifies tweets into two categories: the category of sentimental tweets or subjective tweets (polarity classification) that express a sentiment or an opinion about something and it is classified into two classes (positive or negative), and the category of factual tweets or objective tweets (no sentiments) that do not express any sentiment or opinion, and it is classified into one class (neutral). The second approach classifies tweets into only two classes (positive or negative) which called a **polarity classification** (sentimental tweet).

4.1 First proposed approach

As we said earlier, the first approach consists in classifying the tweets according to three classes (positive, negative and neutral); the proposed idea is that we calculate the semantic similarity between the tweet and three opinion documents (positive document Dp, negative document Dn and neutral document Dne); each document represents a class, that is each document contains words that represent a class; for example, the document of the positive class contains the words: positive, good, happy, etc., and the document of the negative class contains the words: negative, bad, sad, etc.

This proposed approach consists in making a hybrid method based on the notions of the information retrieval system and the semantic similarity using WordNet dictionary and **Leacock and Chodorow** approach. The principle is like we want to calculate the semantic similarity between a user need (request) and a set of documents for finding the ones that are relevant to the request from a point of view of information retrieval systems (IRS), considering in our case the tweet to classify as request and the three opinion documents as the database

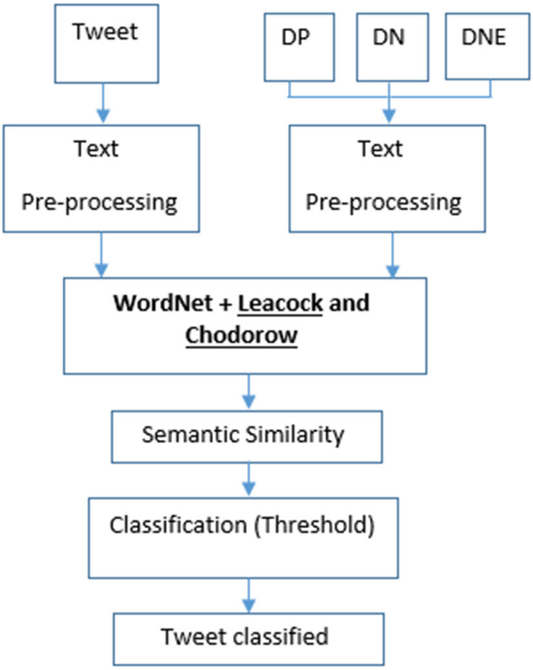


Fig. 3 Classification steps for the first approach

of documents in IRS. For the calculation of the semantic similarity, we use WordNet and the **Leacock and Chodorow** approach.

For classifying a tweet, we calculate the semantic similarity between it and each opinion document (Dp, Dn and Dne). And the tweet takes the class of the document that has the greatest value of the semantic similarity with it.

The following formula shows how we classify the tweets according to the first proposed approach.

$$\text{CV} = \max\left[Sim_{LC}(t, d_p), Sim_{LC}(t, d_n), Sim_{LC}(t, d_{ne})\right] \tag{8}$$

where

- CV is the classification value (positive, negative, neutral).
- $Sim_{LC}(t, d_p)$ is the semantic similarity between the tweet to classify and one of the three documents using the **Leacock and Chodorow** approach.
- t is the tweet to classify.
- d_p, d_n, d_{ne} represent, respectively, the document of the positive class, the document of negative class and the document of the neutral class.

Figure 3 shows the different steps to classify the tweets using our first proposed approach.

DP, DN and DNE represent the three documents that represent the three classes.

According to Fig. 3, the first step is the collection of tweets to analyse (classify); in our work we used two methods to collect the tweets: the first is a Twitter API for java called **Twitter4j**[2] which gives us the ability to collect user's tweets; this API gives us a lot of

[2] Twitter4J (https://twitter4j.org/) is an unofficial Java library for the Twitter API.With Twitter4J; you can easily integrate your Java application with the Twitter service. Twitter4J is an unofficial library.

functionalities which help us to classify tweets such as the possibility of filtering tweets by time or by a given hashtag if we want to analyse for example the tweets of a specific hashtag that represents a product of a company and in a specific time (for example, tweets related to iPhone product and published between 2015 and 2017).

The second method for collecting tweets is Apache Flume; Apache Flume is a tool/service/data ingestion mechanism for collecting aggregating and transporting large amounts of streaming data such as log files, events from various sources to a centralized data store. Flume is a highly reliable, distributed and configurable tool. It is principally designed to copy streaming data (log data) from various web servers to HDFS.

Apache Flume gives us the possibility to collect a large volume of data such as tweets and store them in HDFS (Hadoop distributed file system). In our case, we use Flume to facilitate the collection of tweets and to store them directly in HDFS for analysing them after.

For both methods of collecting the tweets (Twitter4j API or Apache Flume), we need to create a twitter application in the twitter development space.[3] And after configuring it and creating parameters that will then be used by Twitter4j and Flume [25].

The second step after the collection of tweets is the creation of the three opinion documents; each document is a text file that contains words representing a class, that is to say for the document of the positive class we put in it words such as happy, good, happiness, kindness.

The third step is the application of the text preprocessing methods. Before classifying a tweet, it is important to do some process on it in order to facilitate the classification and to reduce the noise exists in it; these treatments are called **text preprocessing**.

Text preprocessing consists of eliminating or transforming a content of the tweet to reduce its noise and to facilitate the classification phase; in the literature we find several types of text preprocessing, namely replacing negative mentions, removing URL, reverting words that contain repeated letters to their original English form, removing numbers, removing stop words and expanding acronyms to their original words by using an acronym dictionary; also we find some works that use the emoticons. In our work we use a lot of them namely:

- **Tokenization:** the phase of splitting the tweet into terms or tokens by removing white spaces, commas and other symbols. It's an important step because in our work we focus on individual words to calculate the semantic similarity between them.
- **Removing Stop Words:** Remove the stop words such as the preposition, are, is, am and the article (a, an, the), etc. The stop words do not emphasize any emotions, so it is important to delete them to reduce the noise from the tweets.
- Remove the Twitter notations such as hashtags (#), retweets (RT), and account Id (@).
- **Removing URL :** the URLs have no effect on the classification so it is important to eliminate them.
- **Removing numbers :** that not express any emotions or attitudes. In general, numbers are no use when measuring sentiment and are removed from tweets to refine the tweet content.
- **Stemming :** Stemming is another very important process; it is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form—generally a written word form. In our work because we focus on the English language, we use the PORTER stemming [26].
- **Effect of negation:** we use a list of words which express the negation such as not, do not, will not, never, cannot, does not; after classification if the tweet is positive or negative, then we use this type of preprocessing text; the idea is that if the tweet, for example, is positive but contains a negation word, then it will be negative and vice versa.

[3] https://apps.twitter.com/.

As presented in Fig. 3 after the text preprocessing step, it is the step of calculating the semantic similarity between the tweet to classify and each class (document) using WordNet and **Leacock and Chodorow** approach, and after the calculation of the semantic similarity the tweet will take the class of the document that has the greatest value of the similarity with it.

This first approach takes advantage of the information retrieval system notions and the best results obtained by the **Leacock and Chodorow** approach as presented previously without forgetting that this approach uses the WordNet which gives good results for the calculation of the semantic similarity.

4.1.1 Example of classification

Our proposed approach can be used in a lot of our daily life scenarios, for example for predicting an election result or for analysing and extracting opinions about a product or a brand before buying it. For example, before buying a smartphone we extract all the tweets related to it from Twitter and then using our proposed method we classify them for knowing the idea of other peoples about this product. Our proposal can also be used by companies to answer questions like: "why a product selling is low?," "have users need are satisfied using our services?" or "are our users happy or want more?".

We assume we want to classify the tweet that says: "I am happy and motivated," using the first proposed approach; as presented earlier the first thing to do is the text preprocessing for eliminating the noise from the tweet. So after that, the tweet becomes: "happy, motivated." The next step is the creation of the three documents that represent the three classes D_p for the class positive, D_n for the class negative and D_{ne} for the class neutral. And after that, it is the step of the calculation of the semantic similarity between the tweet and each document.

$$\begin{cases} S_p = Sim_{LC}(tweet, D_p) \\ S_n = Sim_{LC}(tweet, D_n) \\ S_{ne} = Sim_{LC}(tweet, D_{ne}) \end{cases}$$

After we seek the maximum of these three measures, the tweet takes the feeling of the greatest measure; for example, if S_p is the maximum, the tweet is positive.

4.2 Second proposed approach

The second approach consists of classifying the tweets according to two classes: positive and negative (polarity classification); for that we propose a new function which allows to classify the tweets based on the semantic similarity with the use of WordNet and the approach of **Leacock and Chodorow**; the idea of this new approach is that for each word of the tweet and after the application of the different text preprocessing methods we calculate the difference between the sum of the semantic similarity between each word of the tweet and the word "positive," and the sum of the semantic similarity between each word of the tweet and the word "negative" as presented in the following formula:

$$CV = \sum_{i=1}^{N} Sim_{LC}(W_i, positive) - \sum_{i=1}^{N} Sim_{LC}(W_i, negative) \tag{9}$$

where

- N is the number of words in the tweet.
- CV is the classification value.

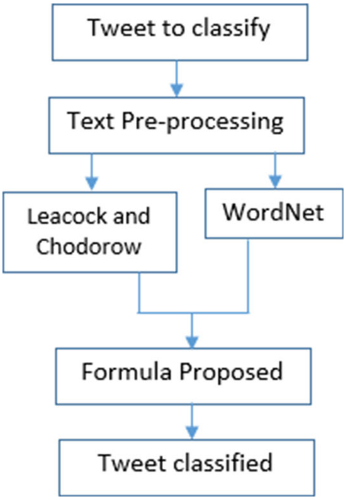


Fig. 4 Classification steps for the second approach

- W_i is the word number i in the tweet.
- $Sim_{LC}(W_i, positive)$ is the semantic similarity using the **Leacock and Chodorow** approach between the word i of the tweet and the word "positive."

For each tweet to classify, the first thing to do is the text preprocessing in order to remove the noise exists in it; after the text preprocessing step, for each word of the tweet we apply the proposed formula. And if the value of the formula after the calculation is positive (greater than 0), then the tweet is positive, otherwise, and if its value is negative (less than 0) the tweet is negative, as presented in the following formula:

$$Sentiment = \begin{cases} \text{“}positive'' & \text{si } CV > 0 \\ \text{“}negative'' & \text{si } CV < 0 \end{cases}$$

The main idea of this approach is to calculate the positivity and the negativity of the tweet. To calculate the positivity, our proposition consists of calculating the semantic similarity between each word of the tweet and the word "**positive**" and then calculating the sum of these semantic similarities to find the value of the tweet's positivity (the first sum of the formula 9). For the negativity, it is the same procedure but this time we calculate the semantic similarity between each word of the tweet and the word "**negative**" (the second sum of the formula 9). After for classifying the tweet, we calculate the difference between the positivity and the negativity and if it is positive(greater than 0) the tweet is positive, and if it is negative(less than 0) the tweet is negative. The procedure is like we compare the values of positivity and negativity and the tweet take the class of the biggest of them.

Compared with the first approach, this approach has the advantage of the complexity and calculation time because it is not necessary to calculate the semantic similarity between each word of the tweet and each word from the three opinion documents (as in the first approach), but it is only between two words "positive" and "negative" which optimise the complexity and the calculation time, and also this approach takes advantage of the WordNet and the approach of **Leacock and Chodorow**.

Figure 4 shows the different steps for classifying a tweet with our second proposed approach.

From Fig. 4 the first thing to do is the collection of tweets to classify using the Twitter4J API or Apache Flume, after that it is the step of the text preprocessing, then with the application of the WordNet and the **Leacock and Chodorow** approach we calculate the semantic similarity score using our proposed formula which going to help us classifying the tweet into two classes: positive or negative.

4.2.1 Example of classification

As with the first approach, the second proposed approach can be used in a lot of domains, for example for helping companies to improve the quality of their products, or in an e-learning platform for helping teachers to analyse the motivation of their students.

We assume we want to classify the same tweet as before that says: "I am happy and motivated"; using the second proposed approach, as presented earlier the first thing to do is the text preprocessing for eliminating the noise from the tweet. So after that, the tweet becomes:"happy, motivated." That is to say its size becomes 2 so $N = 2$.

The next step is the application of our proposed formula to find the sentiment (class) of the tweet:

$$\begin{aligned} CV &= \sum_{i=1}^{N} Sim_{LC}(W_i, positive) - \sum_{i=1}^{N} Sim_{LC}(W_i, negative) \\ &= Sim_{LC}(huppy, positive) + Sim_{LC}(motivated, positive) \\ &\quad - Sim_{LC}(huppy, negative) - Sim_{LC}(motivated, negative) \end{aligned}$$

after the calculation of the classification value, if CV is positive (the positivity is greater than the negativity) the tweet is positive, and it is negative otherwise (the negativity is greater than the positivity).

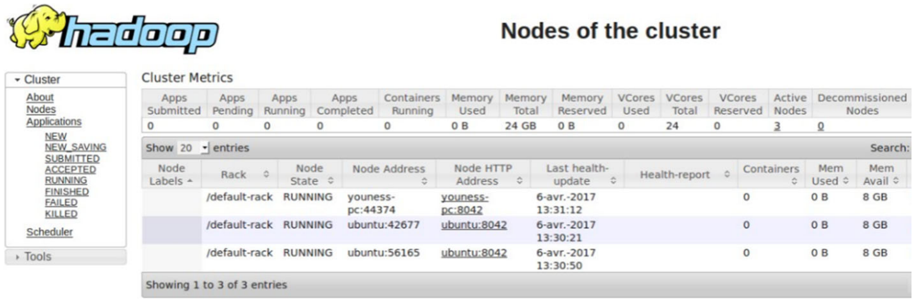
5 Parallelization of the classification

One of the problems that arise when classifying tweets is the time to wait for having the classification result if we have a large tweets database.

To overcome this problem, we have decided to parallelize the work of the classification of tweets with our two proposed methods by working in a Big Data system, using the Hadoop framework by sharing the classification between several machines (Hadoop cluster), using the distributed file system HDFS (Hadoop Distributed File System) for the storage of tweets to classify and also for storing the result of classification, and the MapReduce programming model for parallelization and development of our approaches.

Note that our goal for parallelization is not to study a Hadoop cluster but just to describe how we can parallelize our work using Hadoop and more precisely Hadoop MapReduce, for this we work with Hadoop 2.6.0 with MapReduceV2 (Yarn). The installation of the cluster is done in an operating system Ubuntu 16.04 which will act as the main machine of the cluster (master) and other two Hadoop nodes installing in a virtual machine VMWare workstation 12, so according to this information we can conclude that our goal is parallelization and is not the study of the performance of the Hadoop cluster.

Figure 5 shows the configuration of our cluster (Hadoop machine set) which contains three Hadoop machines, one master machine and two slave machines, so our work will be shared between three machines to reduce the computing time. The nodes used are three nodes with the Ubuntu operating system.



Cluster Metrics

Apps Submitted	Apps Pending	Apps Running	Apps Completed	Containers Running	Memory Used	Memory Total	Memory Reserved	VCores Used	VCores Total	VCores Reserved	Active Nodes	Decommissioned Nodes
0	0	0	0	0	0 B	24 GB	0 B	0	24	0	3	0

Node Labels	Rack	Node State	Node Address	Node HTTP Address	Last health-update	Health-report	Containers	Mem Used	Mem Avail
	/default-rack	RUNNING	youness-pc:44374	youness-pc:8042	6-avr.-2017 13:31:12		0	0 B	8 GB
	/default-rack	RUNNING	ubuntu:42677	ubuntu:8042	6-avr.-2017 13:30:21		0	0 B	8 GB
	/default-rack	RUNNING	ubuntu:56165	ubuntu:8042	6-avr.-2017 13:30:50		0	0 B	8 GB

Showing 1 to 3 of 3 entries

Fig. 5 Configuration of our cluster

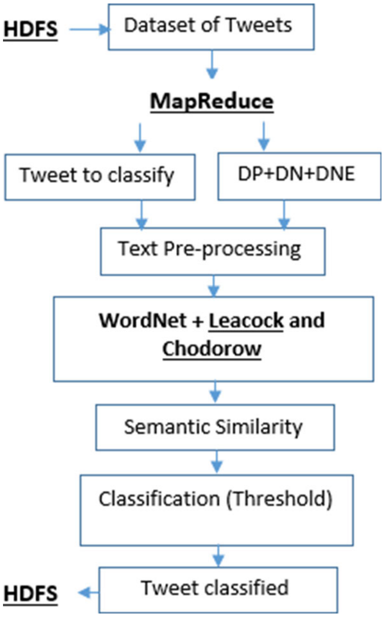


Fig. 6 Parallelization steps for the first approach

5.1 Parallelization steps

5.1.1 First approach

Figure 6 shows the different steps to parallelize our first method using HDFS and Hadoop MapReduce.

From Fig. 6 the first step for parallelization is to store the dataset of tweets to classify in HDFS to share the storage between several machines (Hadoop cluster); in this step we use the Twitter4j API and Apache Flume to collect the tweets, and after the text preprocessing step, it is the step of classification by applying our method but this time with the MapReduce programming model (in a parallel manner).

The input of the MapReduce operation at each iteration (for each tweet) contains a tweet to classify, and the output contains the classified tweet; the classification is done with the application of our first proposed approach; the result of the classification is stored in HDFS. At the end of the classification process, we store the result in HDFS as two columns, one for the tweet as the key of the MapReduce algorithm and the other for the tweet's class as value (positive, negative or neutral).

Our MapReduce algorithm followed for the classification of the tweets with the first approach is presented in Algorithm 1.

Algorithm 1 MapReduce programming model: Approach 1

```
Require: tweet's class
  Cp ← 0
  Cn ← 0
  Cne ← 0
  tweet ← TextPreProcesssing(tweet)
  SE[] ← Split(tweet)
  for all word ∈ SE do
     for all W1 ∈ FilePositive do
        Cp ← Cp+$Sim_{LC}(word, W1)$
     end for
     for all W2 ∈ FileNegative do
        Cn ← Cn+$Sim_{LC}(word, W2)$
     end for
     for all W3 ∈ FileNeutral do
        Cne ← Cne+$Sim_{LC}(word, W1)$
     end for
  end for
  if max(Cp,Cn,Cne)=Cp then
     sentiment ← positive
  else if max(Cp,Cn,Cne)=Cn then
     sentiment ← negative
  else if max(Cp,Cn,Cne)=Cne then
     sentiment ← neutral
  end if
  write(tweet,sentiment)
```

where

- TextPreProcesssing(tweet) is a function that applies the different text preprocessing methods on the tweets.
- Split(tweet) allows us to split the tweet into words, to facilitate its use for calculating the semantic similarity because we focus in our work on individual words.
- $Sim_{LC}(word, W1)$ is a function that calculates the semantic similarity between the tweet to classify and one of the three documents using the **Leacock and Chodorow**.
- max(Cp, Cn, Cne) is a function that returns the maximum of the three measures.
- *write(tweet, sentiment)* is a function that allows storing the result of classification in HDFS in the form of key/value, the tweet to classify as the key of the MapReduce function and the class of the tweet (the result of classification) as value.

5.1.2 Second approach

Figure 7 shows the different steps to parallelize our second method using HDFS and Hadoop MapReduce.

Like the first approach, the first thing to do is the storage of the tweets to classify in HDFS using either Twitter4j API or Apache Flume, and after the storage step it is the step of

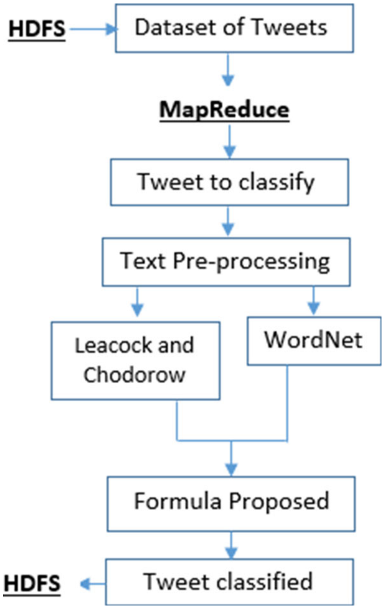


Fig. 7 Parallelization steps for the second approach

classification using our second proposed approach with the MapReduce programming model (in a parallel manner).

The input of the MapReduce at each iteration contains a tweet to classify, and after the application of the different text preprocessing methods and our proposed approach, we classify the tweets into two classes (positive or negative). At the end of the classification process, we store the result in HDFS as two columns, one for the tweet as MapReduce's key and the other for the tweet's class as MapReduce's value.

Our MapReduce algorithm followed for the classification of the tweets with the second approach is the following:

Algorithm 2 MapReduce programming model : Approach 2

```
Require: tweet's class
  S ← 0
  $S_1 \leftarrow 0$
  $S_2 \leftarrow 0$
  tweet ← TextPreProcesssing(tweet)
  SE[] ← Split(tweet)
  for all word ∈ SE do
    $S_1 \leftarrow S_1 + Sim_{LC}(word, positive)$
    $S_2 \leftarrow S_2 + Sim_{LC}(word, negative)$
  end for
  S ← $S_1$ - $S_2$
  if $S > 0$ then
    sentiment ← positive
  else if $S < 0$ then
    sentiment ← negative
  end if
  write(tweet,sentiment)
```

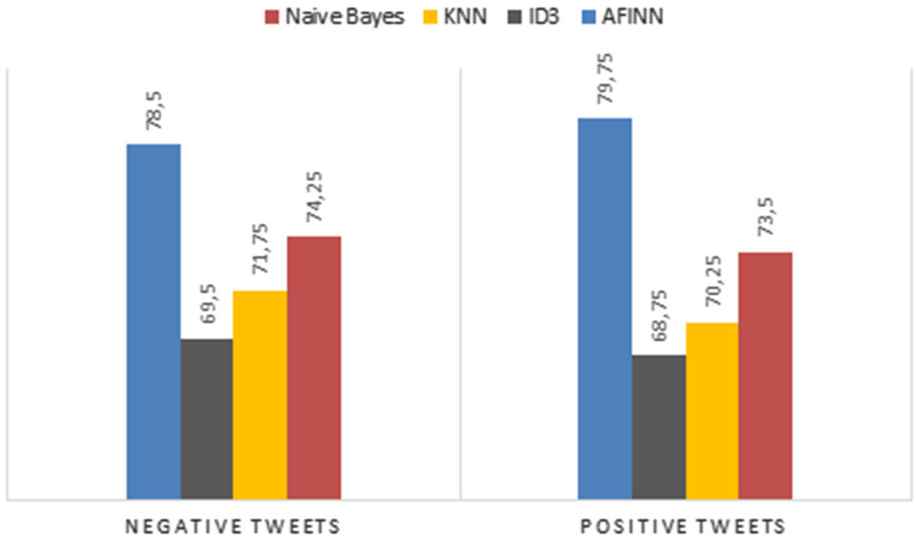


Fig. 8 Precision rate using AFINN dictionary and machine learning algorithms

6 Experiment results

In this section, we will present experimental results of our work by comparing the results obtained using our two approaches with other approaches like with the use of the simple AFINN dictionary and with the use of the AFINN and WordNet (enriching AFINN with semantic)[3].

For that we compute the classification rate, error rate, precision, recall and F1 score of the classification of the tweets using the two proposed approaches and comparing them with the approaches that use simple AFINN with and without WordNet, we also present in this section the effect of the text preprocessing methods on the classification and how methods like removing stopwords, stemming or effect of negation can improve the quality of our two proposed approaches.

The classification will be done in a parallel way using the Hadoop framework with HDFS and the MapReduce programming model of a dataset that contains tweets from different topics collected by Twitter4j API and Apache Flume. These two tools allow us to collect tweets from Twitter easily as presented earlier, for example, collecting tweets related to an event, an election, products or brands in general. And also we can collect tweets that are published in a specific period of time for example between 2015 and 2017.

Most of the sentiment analysis approaches exist in the literature use the machine learning algorithms for classifying tweets as presented in the section of literature review; for that and before comparing our approaches with the approaches based on AFINN, we make an experience which consists of comparing the results obtained with the use of the well-known machine learning algorithms and AFINN for classifying a database of tweets that contains 400 positive and 400 negative tweets; these 800 tweets have randomly chosen from a dataset called Sentiment140, available at http://help.sentiment140.com/for-students. Figure 8 shows the result of the precision rate after the application of three well-known machine learning algorithms and the AFINN dictionary.

Table 2 Effect of text preprocessing on the classification

Methods	With text preprocessing	Without text preprocessing
AFINN	15	26
AFINN+WordNet	10	15
First proposed approach	5	10
Second proposed approach	4	9

According to Fig. 8, for both the negative and the positive tweets AFINN dictionary outperforms the other three machine learning algorithms with a high precision rate (78.5% for the negative tweets and 79.75% for the positive tweets). These results demonstrate why we decided to compare our two proposed approaches with approaches based on AFINN for demonstrating and showing the quality of our methods.

For demonstrating the effect of the application of the text preprocessing methods on the classification and how they can improve the quality of it, we made an experiment that consists of calculating the number of tweets badly classified after the classification with and without the use of text preprocessing methods using our two proposed approaches and with the two approaches that use AFINN dictionary (without and with WordNet). Table 2 shows the results obtained.

According to Table 2, and by using all the approaches, the number of tweets badly classified goes down when we use the text preprocessing methods. (For example, using our first proposed approach the number of badly classified tweets is 10 without text preprocessing but after the use of it this number decreased to 5.) From that, we notice that the use of text preprocessing methods (removing stopwords, stemming, the effect of negation, etc.) has an effect on the classification; it allows to decrease the noise exists in the tweets and the number of tweets badly classified, that is, the text preprocessing increases the quality of the classification.

Another remark from Table 2 is that the number of badly classified tweets decreases by using our two proposed approaches compared with AFINN and WordNet (from 15 and 10 to 5 and 4). That is to say, our two methods outperform the approaches that use AFINN.

From this results, we calculate the classification rate and the error rate for the proposed approaches in comparison with AFINN with and without the use of WordNet using the following Formulas:

$$\mathrm{CR} = \frac{\text{Number of tweet well classified}}{\text{Total number of tweets}} \tag{10}$$

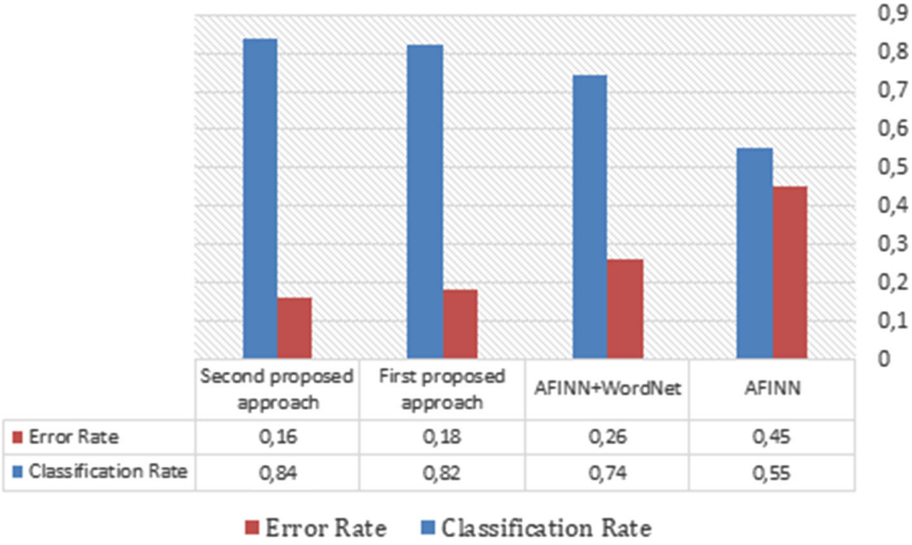
$$\mathrm{ER} = 1 - \mathrm{SR} \tag{11}$$

- CR: classification rate
- ER: error rate

Figure 9 shows the results obtained for CR and ER using our proposed approaches; we compare the results obtained with the approaches of simple AFINN (without WordNet) and AFINN with WordNet, without applying any text preprocessing method.

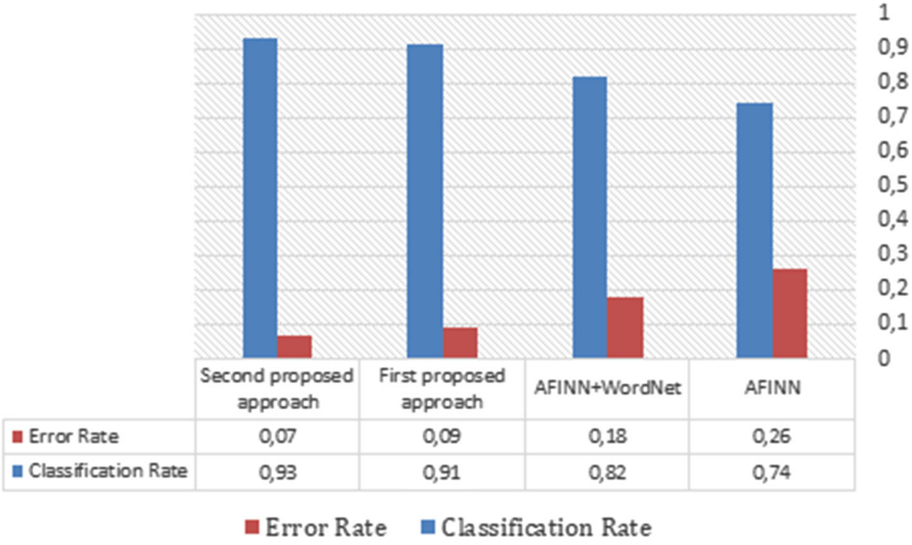
So from Fig. 9, we notice that our methods outperform the method of using the AFINN dictionary with and without semantics (WordNet), with a high classification rate (0.84 and 0.82). Also, our two approaches decrease the error rate (0.16 and 0.18).

To demonstrate the effect of text preprocessing on the classification of tweets, we propose to repeat the same experiment as in Fig. 9 but this time with the use of different text



	Second proposed approach	First proposed approach	AFINN+WordNet	AFINN
🟥 Error Rate	0,16	0,18	0,26	0,45
🟦 Classification Rate	0,84	0,82	0,74	0,55

Fig. 9 Classification and error rate without text preprocessing



	Second proposed approach	First proposed approach	AFINN+WordNet	AFINN
🟥 Error Rate	0,07	0,09	0,18	0,26
🟦 Classification Rate	0,93	0,91	0,82	0,74

Fig. 10 Classification and error rate with text preprocessing

preprocessing methods. Figure 10 shows the results obtained for the classification and error rate with the text preprocessing methods.

From Fig. 10 and as a comparison with Fig. 9, we remark that the text preprocessing methods improve the quality of the classification, that is they increase the classification rate (0.93 and 0.91) and decrease the error rate (0.07 and 0.09), without forgetting that our two proposed methods outperform the approaches of AFINN with and without with a classification equal to 93 and 91%.

For evaluating a sentiment classification system, the CR and ER are not enough, the most commonly used measurements in the evaluation of SA is the four indices: accuracy, recall, precision and the F1 score given by the following formulas:

- Accuracy: the portion of all true predicted instances against all predicted instances.

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{12}$$

- Precision: the portion of true positive predicted instances against all positive predicted instances.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{13}$$

- Recall: the portion of true positive predicted instances against all actual positive instances.

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{14}$$

- F1-score: a harmonic average of precision and recall.

$$\text{F1-score} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{15}$$

where

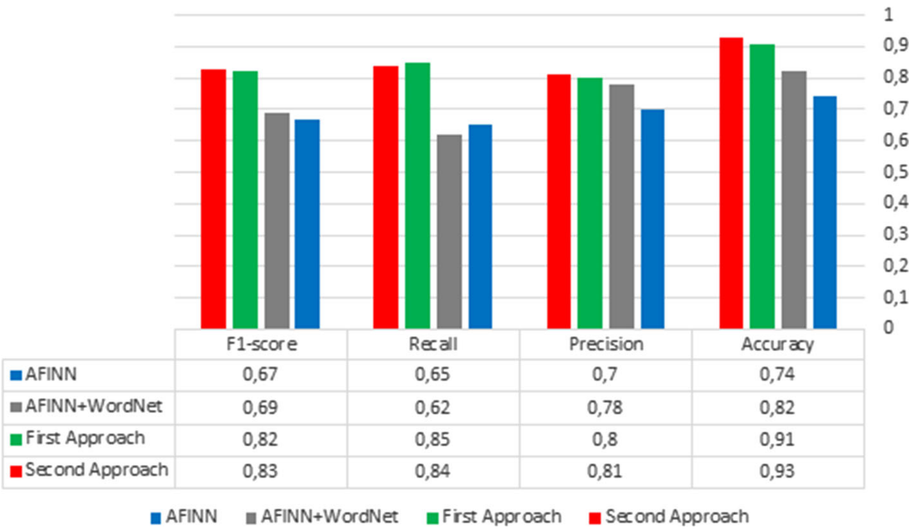
- **TP** (true positives): the correctly predicted positive values which means that the value of the tweet's class is positive and also the predicted value after the classification is positive.
- **TN** (true negatives): the correctly predicted negative values which means that the value of the tweet's class is negative and also the predicted value after the classification is negative.
- **FP** (false positives): when the tweet is negative but the predicted class after the classification is positive.
- **FN** (false negatives): when the tweet is positive but the predicted class after the classification is negative.

For calculating these four parameters, we need to construct a dataset of classified tweets; for that we have used two datasets that contain classified tweets: the first one we have utilized is Sentiment140, available at http://help.sentiment140.com/for-students, and the second dataset, provided by Twitter Sentiment Analysis Training Corpus, which can be downloaded at http://thinknook.com/twitter-sentiment-analysis-training-corpusdataset-2012-09-22/. We have randomly chosen 1000 tweets of each type (negative, positive or neutral) that have been used in our experiments. Figure 11 shows the results obtained after the classification of these tweets using our two proposed approach and the approaches based on AFINN dictionary.

From Fig. 11 we notice that our two proposed methods give good results for the 4 indices (accuracy, precision, recall and F1-score) in comparison with approaches based on AFINN dictionary. The two proposed methods have a good accuracy (93 and 91%), and also they improve the precision as 80 and 81%, the recall as 85 and 84% and the F1-score as 82 and 83%.

Another experiment of our work is to show the effect of the parallelization on the computation time for the classification using our approaches. For that, we have used three Hadoop nodes and a dataset that contains 45,640 tweets. Figure 12 shows the results obtained.

From Fig. 12 we note that if the number of the Hadoop nodes increases the calculation time of the classification with both the first approach proposed and the second approach



	F1-score	Recall	Precision	Accuracy
🟦 AFINN	0,67	0,65	0,7	0,74
⬛ AFINN+WordNet	0,69	0,62	0,78	0,82
🟩 First Approach	0,82	0,85	0,8	0,91
🟥 Second Approach	0,83	0,84	0,81	0,93

Fig. 11 Evaluation of our SA system

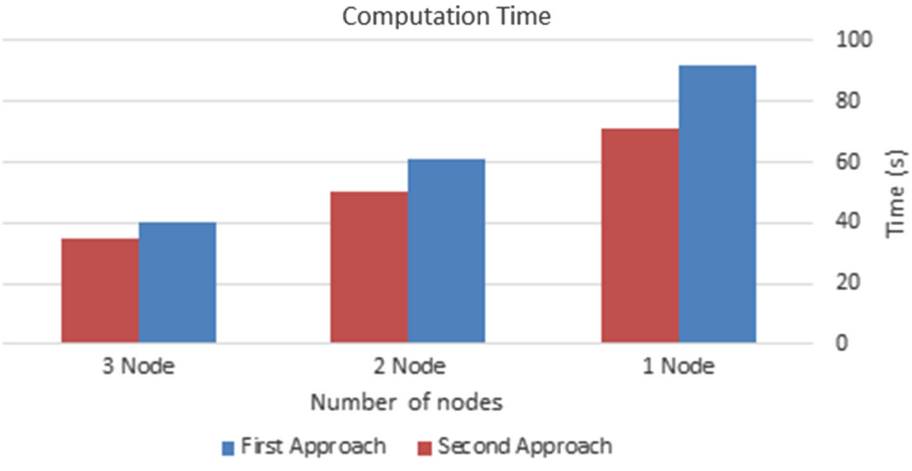


Fig. 12 Computation time of the classification

proposed decreases, which demonstrates the choice of parallelizing the classification. Without forgetting that the computation time necessary to have the result of the classification with the second proposed approach is less than that required with the first approach, this is because the second approach has a few treatments to be done in contrast to the first approach that must go through three documents to have the results of the classification.

So from these results, we conclude that if we have a large number of tweet it is enough to increase the number of Hadoop nodes in order to optimize the classification time. Also, the second approach is better than the first one in the classification time.

7 Conclusion

We have presented in this work our two proposed method to classify the tweets into three classes: positive, negative and neutral (subjectivity classification) with the first approach, and into two classes: positive and negative (polarity classification) with the second approach, our proposition consists in proposing two new methods enriched by semantics using the lexical basis , and we also presented the way to parallelize the classification using Hadoop (HDFS and MapReduce) to optimize the classification time.

The practice shows that our methods give good results at the level of the classification rate, the error rate, precision, accuracy and F1 measure, in addition, the text preprocessing improves the result of classification, and we have also shown the effect of the parallelization on the calculation time.

Our next work lies in this line of research by trying to develop the use of semantics in classification by using graph methods and machine learning algorithms and also the use of sentiment analysis in an adaptive e-learning platform for defining the motivation of learners.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Authors' contributions In this article we present two new approaches for classifying the tweets, proposing two new methods based on the semantic similarity, the notions of IRS and Big Data. All authors read and approved the final manuscript.

References

1. Madani Y, Engourram J, Erritali M, Hssina B, BIRJALI M (2017) Adaptive e-learning using genetic algorithm and sentiments analysis in a big data system. Int J Adv Comput Sci Appl 8(8):394–403
2. Medhat W, Hassan A, Korashy H (2014) Sentiment analysis algorithms and applications: a survey. Ain Shams Eng J 5(4):1093–1113. https://doi.org/10.1016/j.asej.2014.04.011
3. Youness M, Mohammed E, Jamaa B (2017) A parallel semantic sentiment analysis. In: 2017 3rd international conference of cloud computing technologies and applications (CloudTech), pp 1–6
4. Appel O, Chiclana F, Carter J, Fujita H (2016) A hybrid approach to the sentiment analysis problem at the sentence level. Knowl-Based Syst. https://doi.org/10.1016/j.knosys.2016.05.040
5. Shiha MO, Serkan A (2017) The effects of emoji in sentiment analysis. Int J Comput Electr Eng 9(1). https://doi.org/10.17706/ijcee.2017.9.1.360-369
6. Huq MR, Ali A, Rahman A (2017) Sentiment analysis on Twitter data using KNN and SVM. Int J Adv Comput Sci Appl 8(6):19–25
7. Pak A, Paroubek P (2010) Twitter as a corpus for sentiment analysis and opinion mining. In: Proceedings of the seventh conference on international language resources and evaluation, pp 1320–1326
8. Barbosa L, Feng J (2010) Robust sentiment detection on Twitter from biased and noisy data. In: COLING 2010: poster volume, pp 36–44
9. Xie AB, Vovsha I, Rambow O, Passonneau R (2011) Sentiment analysis of Twitter data. In: Proceedings of the ACL 2011 workshop on languages in social media, pp 30–38
10. Jianqiang Z, Xiaolin G (2017) Comparison research on text pre-processing methods on Twitter sentiment analysis. IEEE Access. https://doi.org/10.1109/ACCESS.2017.2672677
11. Haddi E, Liu X, Shi Y (2014) The role of text pre-processing in sentiment analysis. Procedia Comput Sci 17:26–32
12. Saif H, Fernandez M, He Y, Alani H (2014) On stopwords, filtering and data sparsity for sentiment analysis of Twitter. In: Proceedings of the 9th language resources and evaluation conference (LREC), Reykjavik, Iceland, pp 80–81
13. Bao Y, Quan C, Wang L, Ren F (2014) The role of pre-processing in Twitter sentiment analysis. In: Huang DS, Jo KH, Wang L (eds) Intelligent computing methodologies. ICIC 2014. Lecture notes in computer science, vol 8589. Springer

14. Sharif W, Samsudin NA, Deris MM, Naseem R (2016) Effect of negation in sentiment analysis. In: The sixth international conference on innovative computing technology (INTECH 2016). https://doi.org/10.1109/INTECH.2016.7845119
15. Saif H, He Y, Alani H (2012) Semantic sentiment analysis of twitter. In: ISWC'12 (2012) proceedings of the 11th international conference on the semantic web—volume part I, pp 508–524. https://doi.org/10.1007/978-3-642-35176-1_32
16. Saif H, He Y, Fernandez M, Alani H (2014) Semantic patterns for sentiment analysis of Twitter. In: Mika P et al. (eds) The semantic web—ISWC 2014. ISWC 2014. Lecture notes in computer science, vol 8797. Springer, Cham. https://doi.org/10.1007/978-3-319-11915-1_21
17. Tartir Samir, Abdul-Nabi Ibrahim (2017) Semantic sentiment analysis in Arabic social media. J King Saud Univ Comput Inform Sci 29(2):229–233. https://doi.org/10.1016/j.jksuci.2016.11.011
18. Salton G (1989) Automatic text processing: the transformation, analysis, and retrieval of information by computer. Addison- Wesley Publishing Co., Inc., New York
19. Wu Z, Palmer M (1994) Verb semantics and lexical selection. In: Proceedings of the 32nd annual meeting of the associations for computational Llinguistics, pp 133–138
20. Resnik P (1995) Using information content to evaluate semantic similarity in a taxonomy. In: IJCAI, pp 448–453
21. Lin D (1998) An information-theoretic definition of similarity. In: Proceedings of the fifteenth international conference on machine learning (ICML'98). Morgan-Kaufmann, Madison
22. Jiang J, Conrath D (1997) Semantic similarity based on corpus statistics and lexical taxonomy. In: Proceedings of international conference on research in computational linguistics, Taiwan
23. Hirst G, St-Onge D (1998) Lexical chains as representation of context for the detection and correction malapropisms. In: Christiane F (ed), WordNet: an electronic lexical database, chapter 13, pp 305–332. TheMIT Press
24. Leacock C, Chodorow M (1998) Combining local context and WordNet similarity for word sense identification. In: Fellbaum C (ed) WordNet: an electronic lexical database. MIT Press, Cambridge
25. Madani Y, Bengourram J, Erritali M (2017) Social login and data storage in the big data file system HDFS. In: proceedings of the international conference on compute and data analysis, New York, NY, USA, pp 91–97. ACM. https://doi.org/10.1145/3093241.3093265
26. Porter MF (1980) An algorithm for suffix stripping. Orig Publ Program 14(3):130–137