Evaluation of different extractors of features at the level of sentiment analysis

Fatima Es-sabery[1], Khadija Es-sabery[1], Hamid Garmani[1], Junaid Qadir[2], and Abdellatif Hair[1]

***Abstract*—Sentiment analysis is the process of recognizing and categorizing the emotions being expressed in a textual source. Tweets are commonly used to generate a large amount of sentiment data after they are analyzed. These feelings data help to learn about people's thoughts on a various range of topics. People are typically attracted for researching positive and negative reviews, which contain dislikes and likes, shared by the consumers concerning the features of a certain service or product. Therefore, the aspects or features of the product/service play an important role in opinion mining. Furthermore to enough work being carried out in text mining, feature extraction in opinion mining is presently becoming a hot research field. In this paper, we focus on the study of feature extractors because of their importance in classification performance. The feature extraction is the most critical aspect of opinion classification since classification efficiency can be degraded if features are not properly chosen. A few scientific researchers have addressed the issue of feature extraction. And we found in the literature that almost every article deals with one or two feature extractors. For that, we decided in this paper to cover all the most popular feature extractors which are BOW, N-grams, TF-IDF, Word2vec, GloVe and FastText. In general, this paper will discuss the existing feature extractors in the opinion mining domain. Also, it will present the advantages and the inconveniences of each extractor. Moreover, a comparative study is performed for determining the most efficient combination CNN/extractor in terms of accuracy, precision, recall, and F1 measure.**

***Index Terms*—Opinion mining, Extractors of features, Big-Data, Sentiment analysis, text analysis.**

I. INTRODUCTION

With the emergence of the internet and the social networking revolution, a large number of individuals can express freely their views and feelings about entities, products, people, etc. [1, 2]. This growth is accompanied by a huge volume of opinion data available on the web. Indeed, 2.5 billion bytes of data are created every day. In recent years, 90% of the world's data has been generated.

Opinion analysis, in the computer domain, is concerned with the automatic processing of opinions, feelings, and subjectivity expressed or conveyed in textual and audiovisual statements [3]. Opinions concern entities that can be products, services, themes, public persons, organizations, etc. Textual statements can be presented in different formats/types: article in a newspaper, comment/critique in a website post/comment in social networks (Facebook, Twitter, etc.). Oral statements, presented in audiovisual documents, are also presented in different formats: news, radio programs, YouTube videos, etc. This paper focuses on textual statements on Twitter [4].

Twitter is a microblogging service that allows its users to send and read short messages of up to 140 characters [5]. These messages, called "tweets" can be received and sent from your computer or mobile phone. Twitter has only been in existence for five years but has already become a major actor in the social media industry. It is a way of expression of internauts because it permits to exchange in real-time, on all subjects, points of view or needs. These tweets are well suited to the dissemination and propagation of information because they can be republished and also contain hash-tags, that is, tags assigned by the authors of the tweets to briefly characterize the subject of the tweet [6, 7]. Tweets are provided with meta-data as well as information about their location, language, keyword, sentiments expressed, etc.

Several works have been carried out in order to solve the problem of opinion analysis with different methods (linguistic and/or numerical). These works can therefore be classified according to three approaches. The first is symbolic, using lexicons and linguistic rules [8]. The second is a numerical approach based on machine learning methods. Finally, there is a hybrid approach that is a combination of the two previous ones: it uses both lexicons and machine learning algorithms. All these approaches consist in training a classifier based on descriptors, also called features, specific to the opinion analysis task. These features allow us to infer the polarity of a new tweet. Thus, the good performances of the classifiers are conditioned on the one hand by the quantity of training data and on the other hand by the quality of the features. Indeed, the size of the training corpus must be sufficient for training the classifier, and the features must be specific to the task [9, 10].

In general, the opinion process consists of several phases which are the pre-processing stage, the feature extraction stage, the feature selection stage and the classification stage. Feature extraction is considered the most critical step because the performance of the classification depends on the set of extracted features. The choice of features is very important in the performance of the learning model. Generally, the identification of relevant features is done by feature extraction and selection algorithms. Classifier performance varies from one set of

[1]Department of Computer Science, Faculty of Sciences and Technology, Sultan Moulay Slimane University, Beni Mellal 23000, Morocco
(e-mail: {fatima.essabery, khadija.essabery, garmani.hamid}@gmail.com; abd_hair@yahoo.com
[2]Department of Electronics, Quaid-i-Azam University, Islamabad 45320, Pakistan (e-mail: junaidqadirqau@gmail.com).

DOI: 10.36244/ICJ.2022.2.9

features to another. These features require good conception and real thinking to define or guess the right features for the classification task.

Therefore, feature extraction addresses the issue of identifying the most distinguishing, informational, and minimized set of features to enhance the effectiveness of the data treatment. Relevant feature vectors are still the most popular and suitable way of representing the sample for classification issues. Many scientists from various fields, who are focused on data analysis and classification, are working together to address feature extraction challenges. Today's developments in both sentiment analysis and feature extraction algorithms have allowed us to design the identification tools that can accomplish tasks that were previously impossible to do. Feature extraction is at the core of these advancements with applications in sentiment analysis, as well as numerous other developing applications.

For an efficient classification, it is essential to employ an accurate feature extraction approach to retrieve a set of distinguishing and informational features from the input data. In essence, if the retrieved features do not accurately identify the employed signals and are not meaningful, a classification technique employing such set of features may have issues in finding the feature classes labels. Therefore, the classification accuracy may be reduced. Due to the importance of feature extractors, in this paper, we will detail the principle of the most commonly used extractors. The main points of this paper can be summarized as follows:

- The discussion of the existing feature extractors in the opinion mining domain.
- The description of the advantages and the inconveniences of each extractor
- The used dataset is the Sentiment140 dataset contains approximately 1.6 million tweets that were automatically retrieved with the Twitter API.
- Application of multiple feature extractors and determination of the most effective extractor in the case of Sentiment140.
- Implementation of the convolutional neural network, NB, SVM, ID3 and C4.5 as a classifier.
- Setting up the Hadoop framework for the parallel implementation of our proposal

II. RELATED WORKS

Most conventional research papers on sentiment analysis has employed supervised machine learning approaches as the primary module for classification or clustering [11]. These approaches typically exploit the Bag-Of-Words, Word2vec, GloVe, FastText, N-Gram and TF-IDF models to extract the essential features of the text containing user-generated sentiments [12].

A. Baseline feature extraction methods

The authors of the paper [13] evaluated the performance of the feature extractor N-gram in opinion mining field. They proposed to combine the approach based lexicon with the N-gram method for performing the sentiment classification. And their proposed Senti-N-Gram lexicon based approach outperforms well-known unigram-lexicon based method employing the VADER lexicon and an n-gram opinion mining method SO-CAL.

The paper [14] provides an introduction to BoW, its importance, how it operates, its implementations, and the challenges of utilizing it. This review is helpful in terms of introducing the BoW methodology to new researchers and providing a good context with related work to researchers working on this model.

In [15], the authors have analyzed the effect of TF-IDF feature level on the SS-Tweet dataset for opinion extraction. They found that by employing the TF-IDF feature extractor, the sentiment analysis performance is 3-4% higher than by employing the N-gram feature.

The authors of the paper [16], introduce a Word2vec pattern that provides additional linguistic features to accommodate short Chinese dataset. It is compared with the Internet content-based pattern for long dataset. The empirical findings demonstrate that our pattern can effectively improve the performance of opinion classification using six different classes on Weibo.

In the work [17] a hybrid pattern of embedding glove words, contextual and string similarity measures are applied on the large dataset for key sentence retrieval and classification. The empirical findings demonstrate that the GloVe extraction pattern is better than existing metrics for key sentence and string similarity in large datasets.

In [18], the authors have studied the fastText feature extractor and the experimental results show that the FastText achieves 0.97 area under the ROC curve, 94.2% F-measure, and 74.8 ms inference times for CPU.

B. The state-of-the-art feature extractors

The authors of the paper [19] proposed an attention-bidirectional algorithm based on the both deep neural networks CNN and RNN as feature extractor for opinion mining. Their approaches extracted past and future features by taking in consideration the temporal data stream in both senses. In addition, the attention layer mechanism is implemented on the outputs of the bidirectional LSTM layers to emphasize different extracted features to a greater or lesser extent. They also applied the convolution and pooling layers of the CNN in order to reduce the dimensionality of the extracted features. The results of the comparison of their approach and six more recently suggested DNNs for opinion mining indicate that their approach accomplishes the best performance on both short tweet and long review polarities detection.

In the paper [20] a novel efficient method for sentiment classification employing machine learning techniques is suggested. The process of this novel approach is carried out in three phases. In the first phase, the dataset is gathered and pretreated, in the second phase the dataset is tuned by extracting the relevant characteristics, and in the third phase the trained dataset is classified into three classes (negative, neutral, and positive) by implementing several machine learning techniques. Every machine learning algorithms yield distinct results. It is observed that the suggested approach i.e., selective algorithm combined with decision tree provides a high accuracy of 89.47% in comparison to other machine learning techniques

The authors of the paper [21] evaluate different combinations of features in Twitter opinion mining. In addition, they assess and study the effect of combining these separate kinds of characteristics to detect of which aggregation yield crucial insights in the polarity classification task in Twitter opinion mining.

In the paper [22], a comparative study of two extractors (TF-IDF, and Doc2vec) is carried out. The authors of this paper implement these two extractors on three datasets such as Stanford movie review, UCI sentiment, and Cornell movie review datasets. Also, they applied several preprocessing tasks such as removing stop words, eliminating the special characters, stemming and tokenization which increases the accuracy of sentiment classification and reduce the execution of time of used classifier. The pertinent features extracted after the extraction step are tested and trained using various machine learning algorithms like support vector machine, Bernoulli naïve bayes, k-nearest neighbors, decision tree, and logistic regression.

The authors of the paper [23], carried out an experimental analyze of different techniques of feature extraction in Twitter sentiments analysis. Their comparative study is performed in four steps, the first one is the data gathering task which has been carried out from readily available sources. The second phase is the application of several preprocessing tasks utilizing the tool POS. In the third step, various feature selector and extractor are implemented over the collected tweets. Finally, the experimental study is performed for detecting the opinion polarity with different extractors.

Zainuddin et al. [24] proposed a hybrid model for classifying the tweets aspect-based opinion mining. They carried out a comparative analyze in terms of classification rate of three features selectors such as latent semantic analysis, principal component analysis, and random projection. In addition the hybrid model was evaluated employing Twitter datasets to represent various areas, and the evaluation with several machine learning algorithms also proved that the novel hybrid model achieved goods results. Their experimental results showed that the proposed hybrid opinion classification model was capable to increase the classification rate from the existing conventional opinion mining approaches by 76.55%, 71.62% and 74.24 %, respectively.

Pandian suggested in its paper [25], a comparative study of sentiment classification by employing various deep learning models. Its proposed paper has incorporated a feature-extraction with a deep learning model. Furthermore, its research work has three major phases: The first phase is the design of opinion classifiers based on deep learning models. This step is succeeded by the utilization of ensemble techniques and merging of information to get the final ensemble of data sources. As the third phase, an aggregation of ensembles the information is proposed to classify several algorithms along with the suggested algorithm.

III. EXTRACTORS OF FEATURES

Concerning machine learning approaches, many efforts have been performed in the literature on Twitter opinion mining to obtain an efficient vectorization of tweets. In this context, various kinds of features extractors have been suggested already, ranging from simple n-gram based vectorization to meta-level features to word embeddings.

A. N-gram extractor

The N-gram feature extractor is commonly being employed in text based-classification [26]. After applied this extractor, the sentence can be broken down into features of character n-grams and word n-grams. So, an N-gram is a series of “characters or words " picked up, in order, from a body of sentence. N-gram may be unigram (n-gram = 1), bigram (n-gram = 2), trigram (n-gram = 3), and so on.

Usually we pick every word in a sentence to compute the sentiment of the sentence, but there can be a case in which the word is formerly employed in a positive sense, but now it is employed in a negative sense; for example, “what an awesome product, totally waste of money,” if we take only the word “awesome” the sentence will be positive but if we take in consideration the whole sentence, it is indicating the negative. It is because of these types of problems that the N-gram is being developed.

B. TF_IDF extractor

TF_IDF means term frequency - inverse document frequency, which is widely recognized and it is utilized as a weighting procedure and its performance is also still very comparable with new approaches [27]. It is a statistical value that is meant to reflect how much weight a given word has to a certain document in a corpus or a collection. The TF-IDF rate boosts proportionally to the number of occurrences of each term in the document, but is compensated by the occurrence of the term in the corpus, which aids to adapt for the fact that a certain terms occur more frequently in overall. The standardization TF-IDF rates for any document in the corpus via the Euclidean measure are used. The calculations of TF-IDF are presented in the following equation:

$$(\mathrm{TF\_IDF})_{ij} = (\mathrm{TF})_{ij} * \log(\mathrm{IDF})_i \qquad (1)$$

Where $\mathrm{TF} = \frac{k_{ij}}{Number\ of\ the\ words\ in\ the\ sentence}$ with ***k*** is the number of times the word ***i*** appears in the sentence ***j.***

And $\mathrm{IDF} = \frac{Number\ of\ sentences}{Number\ of\ sentences\ with\ the\ word\ i}$

However, the equation 1 is only applied in cases where (TF) ≥ 1. If it does not, TF_IDF = 0, the equation (2) is used.

$$(\mathrm{TF\_IDF})_{ij} \begin{cases} (\mathrm{TF})_{ij} * \log(\mathrm{IDF})_i \ \text{ if } (\mathrm{TF})_{ij} \geq 1 \\ \text{Otherwise} \quad (\mathrm{TF\_IDF})_{ij} = 0 \end{cases} \qquad (2)$$

Where TF denotes the weight standard. It is the weight, which indicates the frequency or relative frequency of the word ***i***, in a given sentence ***j***. And IDF denotes the weight global. It indicates the support of the word ***i*** in respect to ***jth*** belonging to the corpus. In summary:

$(\mathrm{TF})_{ij}$: Number of occurrences of word *i* in sentence *j*.

$(\mathrm{IDF})_{i}$: Number of sentences containing the word *i*.

C. *Bag-of-words extractor*

The bag-of-words is an approach that has been suggested for the first time in the text retrieval area issue for analysis of documents based-text, and it was later induced for computer vision implementations [28]. In general, this approach associates a text with a vector indicating the number of occurrences of each chosen word in the training corpus, For example, we have the three book reviews as presented below:

- **Review A**: This book is very long and boring
- **Review B**: This book is not boring and is shortened
- **Review C**: This book is good and enjoyable

The vocabulary of this three movie reviews consists of eleven words which are: 'This', 'book', 'is', 'very', 'boring', 'and', 'long', 'not', 'shortened', 'good', 'enjoyable'. Therefore the numerical vector of each review is created by the bag-of-word method as follows:

- **Vector of Review A**: [This:1, book:1, is:1, very:1, boring:1, and:1, long:1, not:0, shortened:0, good:0, enjoyable:0]
- **Vector of Review B**: [This:1, book:1, is:1, very:0, boring:1, and:1, long:0, not:1, shortened:1, good:0, enjoyable:0]
- **Vector of Review C:** [This:1, book:1, is:1, very:0, boring:0, and:1, long:0, not:0, shortened:0, good:1, enjoyable:1]

D. *Word2Vec extractor*

Word integration with word2vec [29] identifies the syntactic characteristics of terms and attributes a sentiment score to every term in the vector space. Terms that appear in the identical context are deemed more similar than the terms that appear in the dissimilar contexts. For example, we have a corpus **C** which is composed of a set of tweets, $\boldsymbol{C} = \{\boldsymbol{t_1}, \boldsymbol{t_2}, \boldsymbol{t_3}, \ldots, \boldsymbol{t_n}\}$ and a vocabulary $\boldsymbol{V} = \{\boldsymbol{w_1}, \boldsymbol{w_2}, \boldsymbol{w_3}, \ldots, \boldsymbol{w_m}\}$ is composed of a set of unique words retrieved from **C**. Therefore, the vectorization of the words wi are identified by applying one of the both models Skip-gram or Continuous bag-of-words of Word2Vec in order to compute the probability distribution of the rest words of the set $\boldsymbol{V}\backslash\{\boldsymbol{w_i}\}$ in the context provided by the words wi. In addition, $\boldsymbol{w_i}$ is expressed as a vector space $\boldsymbol{s_i}$ which includes the probabilistic rates of all the other words in the lexicon. The Word2Vec approach extract semantic linked among words in the vocabulary. Furthermore, the obtained set of vectors spaces for all words in the lexicon is high-dimensional and is efficacy for sentiment classification.

E. *GloVe extractor*

The GloVe pattern [30] attempts to create a vector space representation of a term by employing the similarities between the terms as an invariant. The GloVe combines techniques provided by two different patterns, which are the Continuous Bag of Words and Skip-gram pattern. Problem with the former pattern is the low classification rate but its computational time is very efficient, while latter had computational time is inefficient but its classification rate is very high. What the GloVe attempts to do is to integrate the techniques introduced by two patterns and it has demonstrated to be more efficient and accurate than those two patterns.

F. *FastText extractor*

In recent years, Facebook researchers have launched a new word embedding system called FastText [31], which is a quick and effective way to represent each term with vector space and to classify text-based sentiments. The primary goal of fastText term embeddings is to consider the inner structure of terms rather than to learn term representations. FastText operates by Dragging a window over the entry text and either learning the central term from the remainder of the context (by employing the BOW approach), or all the terms in the remainder of the context from the central term by using the Skip-gram approach. The FastText approach is very identical to Word2Vec approach, the only difference is that the FastText learn the vector representation of sub-parts of a term so-called character n-grams.

IV. Advantages and disadvantages of each extractor

In order to implement machine learning approaches to natural language issues, it is necessary to convert the text-based data into digital data. The methods used to carry out this conversion are the extractors described above. Each extractor has the advantages and disadvantages as presented in the tables below:

TABLE I
ADVANTAGES AND DISADVANTAGES OF EACH EXTRACTOR

Extractor	Advantages	Inconveniences
N-gram	-It pick up the representation of the out-of-lexicon terms because it divide the word into N-gram characters [27]. -Simplicity and scalability which means that this approach can efficiently scale small experiments. It can also stock more background with a good understanding of the space-time compromise [28]. -It is efficient in handling textual mistakes and character identification issues that is because of the N-gram structure [27].	- When its parameter N is very large, its parameters space is much too large [28]. - There is also a text smoothness issue because of text sparsity. That's means that we used an approximating function that tries to detect significant features in the data [29]. -It does not take into account the semantics.
Bag-of-words	- It encrypts each term in the lexicon as one-hot vector that renders our training data more meaningful and more expressible, and can be easily rescaled [30]. - It is very simple to comprehend and to implement because is based on one-hot vector representation. - It generates a simplified word representation because it is easier to calculate a likelihood for values by utilizing numerical values. [30].	-It does not take into account the semantics of the term because it does not compute the semantic similarity of each terms [31]. -It does not take into account the semantic connection between the terms because it does not compute the semantic similarity between the terms [31]. -It is suffering from the curse of dimensionality because it represents each term by one-hot vector [30].
TF-IDF	-Short extraction time, simple and easy to calculate because it merges only two notions, term frequency and document frequency [32]. -It provides a certain basic metric to retrieve the most descriptive words in the corpus because it calculates easily the similarities between two sentences or two documents in the corpus [32].	- It does not detect semantics, status in text, co-occurrences in diverse sentences in the corpus because it cannot contribute to convey a semantical sense. [33]. - It is only used as a lexical level characteristic because it gives importance to the terms by the way it weights them and it cannot adequately infer the meaning of the terms and understand their significance in this way [34]. -It cannot pick up the semantics with respect to thematic patterns, term embeddings [34]. - A further drawback is that it may suffer from a lack of memory as TF-IDF may suffer from the curse of dimensionality [33].
Word2Vec	-It identifies the syntactic characteristics of terms because it utilizes a neural network pattern so that once a pattern is trained it can recognize antonymic and synonymous words or can propose a new word to complete an incomplete partial phrase [35].	-It does not learn vectors space of the character n-grams because Word2Vec employs the same vector of numbers to represent any unseen word [35].

TABLE II
ADVANTAGES AND DISADVANTAGES OF EACH EXTRACTOR

Extractor	Advantages	Inconveniences
Word2Vec	-It attributes a sentiment score to every term. For instance, certain negative terms that are adjectives will be more closely related to each other and inversely for positive adjectives. It picks up the semantic and syntactic data of the words [36]. - Its embedding vector size is very small which avoids both drawback of the lack of memory and the curse of dimensionality [36]. -Its context data is never lost because it employs the continuous bag of words method and skip-gram method for predicting the word or the context any word [35].	- It is not very efficacy with term analogy tasks compared to word similarity task [36]. - Word2Vec cannot deal well with out-of-vocabulary terms. It attributes a random vectorial mapping for out-of-vocabulary words, which may be suboptimal [35]. And it is incapable of taking advantage of the statistics of the corpus. - Long extraction time because Word2vec train either continuous bag of words method or skip-gram method and these both methods train the neural network model which trains huge number of instructions and that takes long execution time [36].
GloVe	-It forces term vectors to pick up sub-linear relations in the vector space since vector spaces being by nature linear structures, the easiest way to proceed is to use vector differences [37]. -It outperforms Word2vec in the tasks of terms analogies because it is based on leveraging global word to word co-occurrence counts leveraging the entire corpus [38]. -It adds a more convenient meaning to term vectors by considering the relations between terms pair to pair rather than term to term [37]. -It assigns a smaller weight to very frequent term pairs in order to avoid meaningless terms such as “the”, “a” [38].	-Its pattern is learned on the terms co-occurrence matrix, which requires a lot of storage space because the co-occurrence matrix expands so rapidly and is high-dimensional [38]. -It consumes very time, because the change in level of hyper-parameters requires the reconstruct of the co-occurrence matrix [37]. -It cannot pick up the representation of the out-of-lexicon terms because Glove processes each term in the corpus as an atomic entity and produces a vector for each term [38]. -It is difficult to detach several opposite term pairs using GloVe unlike the Word2Vec [36, 37].
FastText	-It learns usually often the numeric vector of terms in the sentiment analyses process because it is based on the combination of the concept of Word2Vec approach and N-gram method [39]. -It requires a few preprocessing tasks, and little hyper parameter tuning thus needs small memory spaces because it is based on character N-gram [40]. -It learn the vector spaces of character n-grams that make it very efficacy to deal with out-of-vocabulary terms [40].	- Sublinear connections are not explicitly identified because FastText cannot compute the semantic similarity of each term [40]. - As the size of the corpora increases, the memory space used by the FastText word embeddings needs to be increase which takes long execution time for extracting the pertinences features [39]. - It could be very hard to be trained if the Softmax function is used, since the size of the vocabulary is much too big [40].

V. SENTIMENT ANALYSIS METHODOLOGY

In the current work, we train a convolutional neural network as classifier on Sentiment140 dataset [26] of sentiment sentences in order to evaluate each extractor (N-gram, Bag-of-word, TF-IDF, Word2Vec, GloVe, and FastText) for identifying the most efficient one. In general our sentiment analysis methodology consists of four steps which are *data collection phase* in which we used the Sentiment140 dataset, *data pre-processing phase* in which we applied several techniques for improving the data quality and eliminate the data noisy, *feature extraction phase* in which we implemented six extractors in order to determine the most efficient one among them, et finally the *data classification phase*, in which we applied the convolutional neural network (CNN) as classifier as shown in the fig.1.

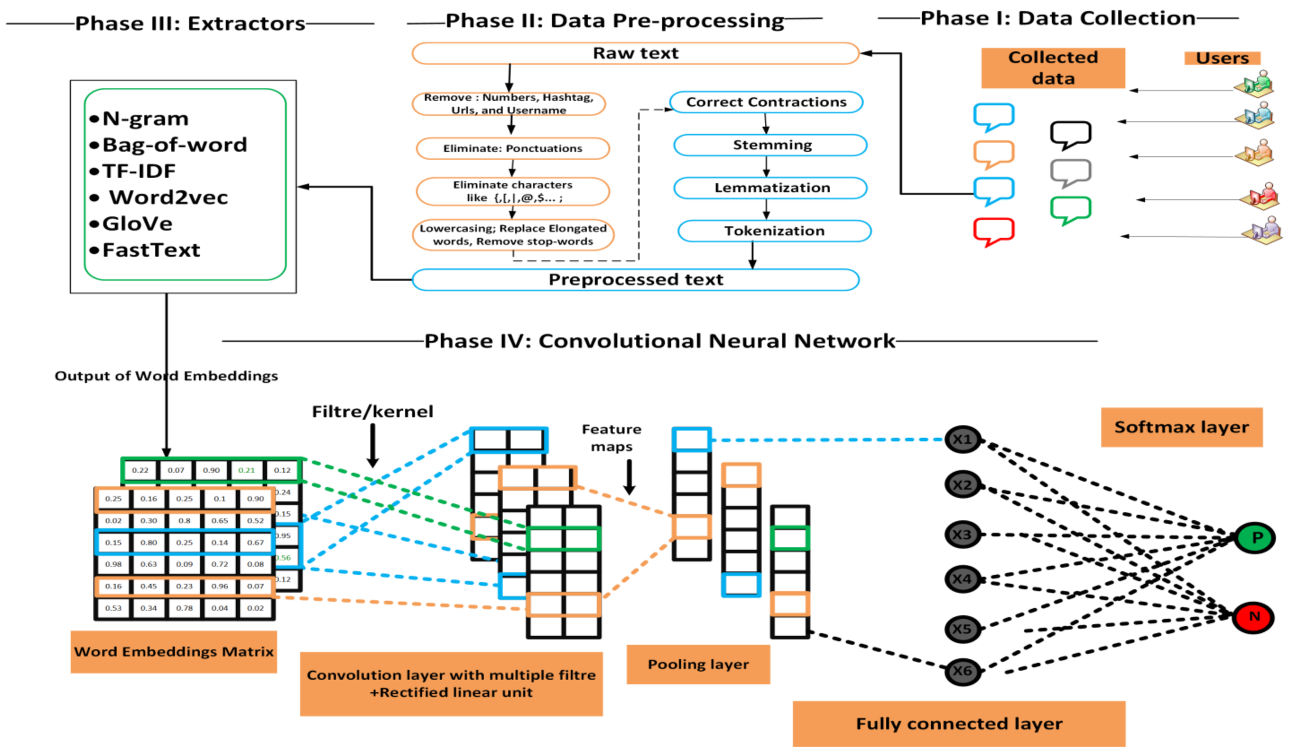

Fig. 1. Architecture global of the sentiment analysis methodology

A. Data collection phase

To implement our contribution we have used the Sentiment140 dataset. This dataset contains approximately 1.6 million tweets that were automatically retrieved with the Twitter API. These tweets were automatically annotated assuming that those containing the ":)" emoticon were positive and those containing the ":(" emoticon were negative. Those containing neither of these emoticons, and those containing both, were not kept. The training set is annotated in two classes (positive and negative) while the test set is annotated by hand on three different classes (positive, negative and neutral). For our experiments, we use only the positive and negative classes of the test set. Table 3 gives the details of the data set.

TABLE 3
DETAILS OF THE USED DATA SET

Training set		Testing set	
Positive	720,000	Positive	80,000
Negative	720,000	Negative	80,000

Each line of the file contains a single tweet with a maximum of 140 characters and can contain several sentences (depending on the length). Because the tweets have been collected directly on the twitter API, they can therefore contain HTML addresses, # hashtags and user names (preceded by @). Finally the structure of each line is as follows:

1. The polarity of the tweet (e.g., 0 = negative, 2 = neutral, 4 = positive).
2. The id of the tweet (e.g., 6532).
3. The date of the tweet (e.g., FAR Sep 18 15:45:31 UTC 2021).
4. The name of the user who posted the tweet (e.g., Essabery).
5. The text of the tweet.

B. Data pre-processing phase

After looking at the data, we saw that the sentences contained HTML tags, empty words and all punctuation. So we started by removing the noise to normalize our sentences. We remove HTML tags with the BeautifulSoup2 module. We also remove all the characters that are not letters and therefore, remove all punctuation from texts. Because stop words, by definition, do not bring any information to the text, we eliminate them too. All letters are also changed to lower case. Finally, we root all the words to process each inflection of a word into a single word. Below we detail some important pre-processing steps.

Nicknames: Since nicknames (e.g., @username) are useless for sentiment analysis, we replace all @usernames with the text AT_USER so that we can delete them later.

Repeated letters: The language used on Twitter is mostly familiar. It is therefore not uncommon for words to be written with a letter (or several) that is repeated when it should not be. For example the word "dog" can be found as "doooooooog" on Twitter. As soon as a word contains identical letters that are repeated more than more than twice, they are replaced by only two occurrences of the same letter ("doooooooog" becomes “doog").

Hashtags: Twitter hashtags are used to create an instant connection with other users. The word that follows the # is usually a word that provides a lot of information about the sentiment of the sentence. We keep this word, but the hashtag character is removed.

Lemmatization: We transform all inflections into their root. The objective is to reduce the derived forms of a word to a common base form in order to facilitate the correspondence between the different terms.

1) *EFFECT OF PRETREATMENT*

Table 4 shows the effect of these preprocessing on the number of useful words in the text.

TABLE 4
EFFECT OF NOISE REDUCTION

Reduction	Number of features	% of the original
None	1 569 914	100%
Username	65 993	96.88%
URLs	609 692	54.22%
Repeated letters	298 673	78.35%
All	984 139	39.43%

All these text noise removals lead to a reduction of the corpus set to 39.43% of the original corpus size.

TABLE 5
ACCURACY AND ERROR RATE WITHOUT AND WITH PREPROCESSING

Criteria	Without preprocessing	With preprocessing
Accuracy	50.19%	82.35%
Error rate	49.81%	17.65%

As shown in the table below (5), the preprocessing tasks reduce the error rate from 49.81% to 17.65 and increase the accuracy from 50.19% to 82.35%. So, it is necessary to apply the preprocessing process before the application of machine learning algorithm.

C. *Feature extraction phase*

In order to obtain a reliable system based on a numerical approach, the design of good features is the most important step for classification. Bags of words, n-grams, TF-IDF, Word2vec, GloVe and FastText are the most common extractor of features in sentiment analysis. The main purpose of this work is to test different sets of extractors and pre-trained word embeddings by applying the CNN classifier.

D. *Data classification phase*

After the feature extraction step, the next step is the data classification in which we have used the CNN. The CNN is a specialized type of multi-layer neural network generally used when the input is structured according to a grid (e.g. an image). These networks were inspired by the visual cortex of animals, and more particularly on its properties: local receptive fields and weight sharing. Figure 2 shows the different layers of a convolutional neural network. The latter is composed of one or more convolution and pooling blocks, one or more hidden layers and an output layer. The CNN takes as input a multi-dimensional grid representing a learning or inference instance, and provides as output the corresponding class.

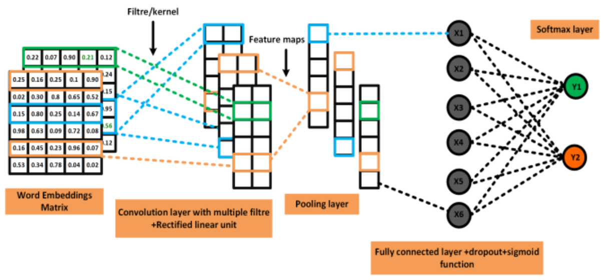
Filtre/kernel
Feature maps
Softmax layer
Word Embeddings Matrix
Convolution layer with multiple filtre +Rectified linear unit
Pooling layer
Fully connected layer +dropout+sigmoid function
X1
X2
X3
X4
X5
X6
Y1
Y2

Fig. 2. Simple version of the convolutional neural network

VI. EXPERIMENTAL RESULTS

As mentioned earlier, this approach consists of evaluating the set of extractors described below using the CNN as a classifier. In this section we will examine the performance of each extractors by applying them on the corpus Sentiment10 and by computing four evaluations criteria [41] which are the following:

Precision (P): represents the average of the precisions of the k classes. It is calculated according to equation (5).

$$P = \frac{\sum_{j=1}^{k} P_j}{k} \quad (5)$$

With:

$$P_j = \frac{\textit{number of sentences correctly assigned to the class } y_i}{\textit{number of sentences assigned to the class } y_i} \quad (6)$$

Recall (R): represents the average of the recalls of the k classes. It is calculated according to the following equation

$$R = \frac{\sum_{j=1}^{k} R_j}{k} \quad (7)$$

With:

$$R_j = \frac{number\ of\ sentences\ correctly\ assigned\ to\ the\ class\ y_i}{number\ of\ sentences\ belong\ to\ the\ class\ y_i} \quad (8)$$

F1 measure (F1): represents the harmonic mean of precision and recall. It measures the performance of the system and is calculated according to equation (9).

$$F1 = \frac{2 \times P \times R}{P + R} \quad (9)$$

Accuracy A: evaluates our approach in an overall [23]. It is calculated according to the equation (10).

$$A = \frac{Number\ of\ sentences\ correctly\ classified}{Total\ number\ of\ sentences} \quad (10)$$

We have also implemented our approach on Hadoop framework with a cluster of five machine: four slave nodes and one master node. The configuration of the cluster is presented in the following table 6:

TABLE 6
PARAMETERS SETTING OF THE HADOOP CLUSTER

Configuration	**Parameters**
N. of nodes	5
apache hadoop	version 2.7.2
OS	Ubuntu 20.04.4
Memory	16 GB
CPU	Intel(R) Core(TM) i7-4810MQ CPU @ 2.80GHz 2.80 GHz

The Parameters settings of CNN used in our implementation are shown in the following table 7:

TABLE 7
PARAMETERS SETTING OF THE CNN

Parameter	Value
Vocabulary size	45,000
Padding	0
Regularizer	L2
Size of filter	4,7
Number of filter	15
Activation function	ReLU
Function of Pooling layer	Max-pooling
N. of pooling layer	3
N. of convolutional layer	3
Input embedding matrix	500x500

A. Impact of the choice of input embeddings on the CNN

The CNN architecture was trained on the Sentiment140 corpus with the different existing pre-trained word embeddings (Bag-of-Word, N-grams, TF-IDF, Word2vec, GloVe, and FastText). The table 3 reports the performance of our approach in terms of precision, recall, F1 measure and accuracy.

TABLE 8
P, R F1, A OF THE ALL COMBINATIONS OF OUR APPROACH

Combination\criteria (%)	P	R	F1	A
CNN+BOW	64.08	63.51	63.79	66.92
CNN+N-grams	49.97	60.23	54.62	58.49
CNN+TF-IDF	68.88	71.59	70.20	70.02
CNN+Word2vec	86.13	83.55	84.82	85.06
CNN+GloVe	79.49	80.05	79.76	79.54
CNN+FastText	**93.43**	**90.89**	**92.14**	**91.32**

From the table 8, we remark that the combination **CNN+FastText** performs better than other combinations in terms of P(93.43%), R(90.89%), F1(92.14%) and A(91.32%).

Pattern complexity is a metric for the time and space consumption used by a pattern. In these following tables, we assessed the time and space complexity of all combinations of our approach.

TABLE 9
SPACE COMPLEXITY OF THE ALL COMBINATIONS OF OUR APPROACH

Combination\ complexity	N. operations	N. parameters
CNN+BOW	65.5M	40M
CNN+N-grams	77.25M	36M
CNN+TF-IDF	89M	46M
CNN+Word2vec	53.5M	29M
CNN+GloVe	45M	27M
CNN+FastText	**39M**	**22M**

As the experimental findings indicate in the table 9 above, the combination **CNN+FastText** requires computational complexity in space much lower than others combinations. Since it carried out numerous operations with a size equal to 39M and its size of parameters is equal to 22M.

The following table 10 illustrates the experimental findings after gauging the computational time complexity of all combinations of our approach in terms of both training time consumed and testing time consumed.

TABLE 10
TIME COMPLEXITY OF THE ALL COMBINATIONS OF OUR APPROACH

Combination\ complexity	Training time	Testing time
CNN+BOW	46.23s	17.5s
CNN+N-grams	51s	19s
CNN+TF-IDF	69s	21s
CNN+Word2vec	38.25s	14.5s
CNN+GloVe	28.65s	15.25s
CNN+FastText	**15.46s**	**10.98s**

As the experimental findings indicate in the table 10 above, the combination **CNN+FastText** requires computational complexity in time much lower than others combinations. Since it consumed a training time equal to 15.46s and a testing time equal to 10.98s.

B. A comparison of our approach with other machine learning algorithms.

This experiment makes a comparison of the combinations of four machines-learning algorithms which are Naive Bayes (NB), Support Vector Machine (SVM), ID3 and C4.5 decision tree algorithm with six feature extractors which are BOW, N-grams, TF-IDF, Word2vec, GloVe and FastText in terms of precision (P), Recall (R), and F1 measure (F1) and Accuracy (A). Its empirical findings are displayed in the Table 11.

TABLE 11
P, R, F1 AND A OF THE COMBINATION OF FOUR MACHINES- LEARNING ALGORITHMS AND SIX FEATURE EXTRACTORS

Combination\criteria	P	R	F1	A
NB+BOW	48.31	45.64	46.93	47.12
NB+N-grams	35.84	34.15	34.97	36.02
NB+TF-IDF	50.97	51.08	51.02	52.68
NB+Word2vec	49.64	50.09	49.86	50.32
NB+GloVe	53.26	55.42	54.31	55.29
NB+FastText	**56.74**	**57.49**	**57.11**	**58.18**
SVM+BOW	45.61	44.88	45.24	46.07
SVM+N-grams	40.33	39.50	39.91	40.26
SVM+TF-IDF	51.64	50.37	50.99	51.63
SVM+Word2vec	45.87	46.25	46.05	45.91
SVM+GloVe	50.89	49.68	50.27	49.82
SVM+FastText	**60.43**	**61.27**	**60.48**	**61.39**
ID3+BOW	62.54	63.19	62.86	63.42
ID3+N-grams	53.48	54.02	53.74	54.13
ID3+TF-IDF	64.85	63.94	64.39	65.07
ID3+Word2vec	58.46	60.32	59.37	61.53
ID3+GloVe	64.15	63.24	63.69	64.18
ID3+FastText	**70.65**	**72.39**	**71.50**	**72.87**
C4.5+BOW	60.58	59.67	60.12	59.93
C4.5+N-grams	58.34	60.59	59.44	61.48
C4.5+TF-IDF	70.49	71.64	71.06	70.97
C4.5+Word2vec	68.31	69.25	68.77	69.51
C4.5+GloVe	71.58	73.82	72.68	73.96
C4.5+FastText	**77.65**	**76.92**	**77.28**	**76.64**

From the table 11, we remark that the combination **machine-learning algorithm+FastText** performs better than other combinations in terms of P, R, F1 and A. Therefore, we notice that the feature extractor FastText outperforms all others feature extractors (BOW, N-grams, TF-IDF, Word2vec, and GloVe). And from the tables 10 and 11, we remark that **CNN+FastText** performs better than others machine learning algorithms (NB,SVM,ID3 and C4.5) in terms of P(93.43%), R(90.89%), F1(92.14%) and A(91.32%).

In Table 11, we see that some values are lower than 0.5. In the case of the NB classifier. This is because the input values to the NB classifier are numerical values in this contribution. And, as we know from the machine learning literature, NB performs well for categorical versus numerical input variables.

C. A comparison of our approach with other approaches selected from the existing literature.

For further testing of our proposed approach, we carried out another experiment aimed at comparing our method with the other approaches taken from the literature, namely Naresh et al. [14], Carvalho et al. [15], Avinash et al. [16], Kumar et al. [17] and Zainuddin et al. [18]. However, in this experiment, the evaluation measures used will be precision (P), Recall (R), and F1 measure (F1) and Accuracy (A). Its empirical findings are displayed in the Table 12.

TABLE 12
P, R, F1 AND A OF OUR APPROACH WITH OTHER APPROACHES SELECTED FROM THE EXISTING LITERATURE

Approach	P	R	F1	A
Naresh et al. [14]	65.48	67.12	66.28	66.87
Carvalho et al. [15]	79.56	80.04	79.79	78.95
Avinash et al. [16]	70.19	69.38	69.78	68.42
Kumar et al. [17]	83.21	82.40	82.80	81.94
Zainuddin et al. [18]	69.34	70.67	69.99	71.68
CNN+FastText	93.43	90.89	92.14	91.32

From the results shown in the table 12, we remark that our approach (CNN+FastText) obtained the strongest performances in terms of accuracy (91.32%), precision (93.43%), recall (90.89%), and F1 measures (92.14%) compared to other chosen classifiers from the literature which are Naresh et al. [14], Carvalho et al. [15], Avinash et al. [16], Kumar et al. [17] and Zainuddin et al. [18].

VII. CONCLUSION

Feature extraction is needed to get good performance in sentiment classification. The purpose of feature extraction is to identify the strongest and most informational set of features to enhance the effectiveness of the classifier. Moreover, Feature extraction is the most critical aspect of opinion classification since classification efficiency can be negatively affected if features are not properly chosen. For that, in this paper, we presented a preliminary study of the most popular feature extractors. And, we combined a CNN, NB, SVM, ID3, and C4.5 with several word embedding methods in order to identify the most efficient extractor of features that positively affected the classifier performances. Accordingly to the experimental results, the performance of the used classifiers varies a little with the nature of the word embedding sets. In general we found the combination **CNN+FastText** outperforms all other combinations in terms of accuracy, precision, recall, and F1 measure.

REFERENCES

[1] R. Ahuja, A. Chug, S. Kohli, S. Gupta, and P. Ahuja, "The Impact of Features Extraction on the Sentiment Analysis," Proceedings of the 2019 International Conference on Pervasive Computing Advances and Applications, Jaipur, India, (2019) January 8-10.

[2] M. S. Neethu and R. Rajasree, "Sentiment analysis in twitter using machine learning techniques," Proceedings of the 2013 Fourth International Conference on Computing, Communications and Networking Technologies, Tiruchengode, India, (2013) July 4-6.

[3] H. Saif, Y. He, and H. Alani, "Semantic Sentiment Analysis of Twitter," Proceedings of the International Conference on Semantic Web, Boston, MA, USA, (2012) November 11-15.

[4] N. Al-Twairesh and H. Al-Negheimish, "Surface and Deep Features Ensemble for Sentiment Analysis of Arabic Tweets," Journal of IEEE Access, vol. 7, (2019), pp. 84122–84131.

[5] M. Venugopalan and D. Gupta, "Exploring sentiment analysis on twitter data," Proceedings of the 2015 Eighth International Conference on Contemporary Computing, Noida, India, (2015) August 20-22.

[6] H. Kaur, V. Mangat, and Nidhi, "A survey of sentiment analysis techniques," Proceedings of the 2017 International Conference on IoT in Social, Mobile, Analytics and Cloud, Palladam, India, (2017) February 10-11.

[7] S. Liao, J. Wang, R. Yu, K. Sato, and Z. Cheng, "CNN for situations understanding based on sentiment analysis of twitter data," Proceedings of the 8th International Conference on Advances in Information Technology, Macau, China, (2016) December 19-22.

[8] B. Gupta, M. Negi, K. Vishwakarma, G. Rawat, and P. Badhani, "Study of Twitter Sentiment Analysis using Machine Learning Algorithms on Python," International Journal of Computer Application, vol. 165, no. 9, pp. 29–34.

[9] H. Hamdan, P. Bellot, and F. Bechet, "Lsislif: Feature Extraction and Label Weighting for Sentiment Analysis in Twitter," Proceedings of the 9th International Workshop on Semantic Evaluation, Denver, Colorado, (2015) June 4-5.

[10] A. P. Jain and P. Dandannavar, "Application of machine learning techniques to sentiment analysis," Proceedings of the 2nd International Conference on Applied and Theoretical Computing and Communication Technology, Bangalore, India, (2016) July 21-23.

[11] F. Es-sabery, K. Es-sabery, and A. Hair, "A MapReduce Improved ID3 Decision Tree for Classifying Twitter Data," Edited M. Fakir, M. Baslam, and R. El Ayachi, Springer, Cham, (2021), pp. 160–182.

[12] F. Es-Sabery et al., "A MapReduce Opinion Mining for COVID-19-Related Tweets Classification Using Enhanced ID3 Decision Tree Classifier," International Journal of the IEEE Access, vol. 9, (2021), pp. 58706–58739.

[13] A. Dey, M. Jenamani, and J. J. Thakkar, "Senti-N-Gram: An n-gram lexicon for sentiment analysis," Expert Systems with Applications, vol. 103, pp. 92–105, Aug. 2018, DOI: 10.1016/j.eswa.2018.03.004.

[14] W. A. Qader, M. M. Ameen, and B. I. Ahmed, "An Overview of Bag of Words; Importance, Implementation, Applications, and Challenges," in 2019 International Engineering Conference (IEC), Jun. 2019, pp. 200–204. DOI: 10.1109/IEC47844.2019.8950616.

[15] R. Ahuja, A. Chug, S. Kohli, S. Gupta, and P. Ahuja, "The Impact of Features Extraction on the Sentiment Analysis," Procedia Computer Science, vol. 152, pp. 341–348, Jan. 2019, DOI: 10.1016/j.procs.2019.05.008.

[16] B. Shi, J. Zhao, and K. Xu, "A Word2vec Model for Sentiment Analysis of Weibo," in 2019 16th International Conference on Service Systems and Service Management (ICSSSM), Jul. 2019, pp. 1–6. DOI: 10.1109/ICSSSM.2019.8887652.

[17] S. Anjali Devi and S. Sivakumar, "An efficient contextual glove feature extraction model on large textual databases," Int J Speech Technol, Sep. 2021, https://doi.org/10.1007/s10772-021-09884-2.

[18] J. Kralicek and J. Matas, "Fast Text vs. Non-text Classification of Images," in Document Analysis and Recognition – ICDAR 2021, Cham, 2021, pp. 18–32. DOI: 10.1007/978-3-030-86337-1_2.

[19] M. E. Basiri, S. Nemati, M. Abdar, E. Cambria, and U. R. Acharya, "ABCDM: An Attention-based Bidirectional CNN-RNN Deep Model for sentiment analysis," International Journal of the Future Generation Computer Systems, vol. 115, (2021), pp. 279–294.

[20] A. Naresh and P. Venkata Krishna, "An efficient approach for sentiment analysis using machine learning algorithm," International Journal of the Evolutionary Intelligence, vol. 14, no. 2, (2021), pp. 725–731.

[21] J. Carvalho and A. Plastino, "On the evaluation and combination of state-of-the-art features in Twitter sentiment analysis," International Journal of the Artificial Intelligence Review, vol. 54, no. 3, (2021), pp. 1887–1936.

[22] M. Avinash and E. Sivasankar, "A Study of Feature Extraction Techniques for Sentiment Analysis," Edited M. Ajith, M. Paramartha, M. Jyotsna, M. Abhishek, M. Soumi, Springer, Cham, (2019), pp. 475–486.

[23] J. A. Kumar and S. Abirami, "An Experimental Study of Feature Extraction Techniques in Opinion Mining," International Journal on Soft Computing, Artificial Intelligence and Applications, vol. 4, no. 1, (2015), pp. 15–21.

[24] N. Zainuddin, A. Selamat, and R. Ibrahim, "Hybrid sentiment classification on twitter aspect-based sentiment analysis," International Journal of the Applied Intelligence, vol. 48, no. 5, (2018), pp. 1218–1232.

[25] A. P. Pandian, "Performance Evaluation and Comparison using Deep Learning Techniques in Sentiment Analysis," International Journal of Soft Computing Paradigm, vol. 3, no. 2, (2021), pp. 123–134.

[26] F. Es-Sabery et al., "Sentence-Level Classification Using Parallel Fuzzy Deep Learning Classifier," International Journal of IEEE Access, vol. 9, (2021), pp. 58706–58739.

[27] F. Es-sabery, K. Es-sabery, H. Garmani, and A. Hair, "Sentiment Analysis of Covid19 Tweets Using A MapReduce Fuzzified Hybrid Classifier Based On C4.5 Decision Tree and Convolutional Neural Network," E3S Web Conf., vol. 297, p. 01052, 2021, DOI: 10.1051/e3sconf/202129701052.

[28] F. Es-sabery and A. Hair, "A MapReduce C4.5 Decision Tree Algorithm Based on Fuzzy Rule-Based System," Fuzzy Information and Engineering, vol. 11, no. 4, (2019), pp. 446–473, DOI: 10.1080/16168658.2020.1756099.

[29] F. Es-sabery and A. Hair, An Improved ID3 Classification Algorithm Based On Correlation Function and Weighted Attribute *. 2019, p. 8. DOI: 10.1109/ISACS48493.2019.9068891.

[30] H. Choi, K. Cho, and Y. Bengio, "Context-dependent word representation for neural machine translation," Computer Speech & Language, vol. 45, pp. 149–160, Sep. 2017, DOI: 10.1016/j.csl.2017.01.007.

[31] L. Wu, S. C. H. Hoi, and N. Yu, "Semantics-Preserving Bag-of-Words Models and Applications," IEEE Transactions on Image Processing, vol. 19, no. 7, pp. 1908–1920, Jul. 2010, DOI: 10.1109/TIP.2010.2045169.

[32] S.-W. Kim and J.-M. Gil, "Research paper classification systems based on TF-IDF and LDA schemes," Hum. Cent. Comput. Inf. Sci., vol. 9, no. 1, p. 30, Aug. 2019, DOI: 10.1186/s13673-019-0192-7.

[33] M. Karthiga, S. Sountharrajan, A. Bazila Banu, S. Sankarananth, E. Suganya, and B. Sathish Kumar, "Similarity Analytics for Semantic Text Using Natural Language Processing," in 3rd EAI International Conference on Big Data Innovation for Sustainable Cognitive Computing, Cham, 2022, pp. 239–248. DOI: 10.1007/978-3-030-78750-9_17.

[34] V. Kalra, I. Kashyap, and H. Kaur, "Improving document classification using domain-specific vocabulary: hybridization of deep learning approach with TFIDF," Int. j. inf. tecnol., Mar. 2022, DOI: 10.1007/s41870-022-00889-x.

[35] P. F. Muhammad, R. Kusumaningrum, and A. Wibowo, "Sentiment Analysis Using Word2vec And Long Short-Term Memory (LSTM) For Indonesian Hotel Reviews," Procedia Computer Science, vol. 179, pp. 728–735, Jan. 2021, DOI: 10.1016/j.procs.2021.01.061.

[36] B. Li, A. Drozd, Y. Guo, T. Liu, S. Matsuoka, and X. Du, "Scaling Word2Vec on Big Corpus," Data Sci. Eng., vol. 4, no. 2, pp. 157–175, Jun. 2019, DOI: 10.1007/s41019-019-0096-6.

[37] J. Bernabé-Moreno, A. Tejeda-Lorente, J. Herce-Zelaya, C. Porcel, and E. Herrera-Viedma, "A context-aware embeddings supported method to extract a fuzzy sentiment polarity dictionary," Knowledge-Based Systems, vol. 190, p. 105236, Feb. 2020, DOI: 10.1016/j.knosys.2019.105236.

[38] M. I. Prabha and G. Umarani Srikanth, "Survey of Sentiment Analysis Using Deep Learning Techniques," in 2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT), Apr. 2019, pp. 1–9. DOI: 10.1109/ICIICT1.2019.8741438.

[39] P. Mojumder, M. Hasan, Md. F. Hossain, and K. M. A. Hasan, "A Study of fastText Word Embedding Effects in Document Classification in Bangla Language," in Cyber Security and Computer Science, Cham, 2020, pp. 441–453. DOI: 0.1007/978-3-030-52856-0_35.

[40] A. G. D'Sa, I. Illina, and D. Fohr, "BERT and fastText Embeddings for Automatic Detection of Toxic Speech," in 2020 International Multi-Conference on: "Organization of Knowledge and Advanced Technologies" (OCTA), Feb. 2020, pp. 1–5. DOI: 10.1109/OCTA49274.2020.9151853.

[41] F. Es-sabery, K. Es-sabery, B. El Akraoui, and A. Hair, "Optimization Focused on Parallel Fuzzy Deep Belief Neural Network for Opinion Mining," in Business Intelligence, Cham, 2022, pp. 3–28. DOI: 10.1007/978-3-031-06458-6_1.

Fatima Es-Sabery received PhD degree in BigData Mining from the Department of Computer Sciences, Sultan Moulay Sliman University, Beni Mellal, Morocco, in 2022. Her general research interests include data mining area, big data field, wireless sensor networks, fuzzy systems, machine learning, deep learning, and the Internet of Things.

Khadija Es-Sabery received the Engineering degree from the Department of Computer Science, National School of Applied Sciences, Cadi Ayyad University, Marrakech, Morocco, in 2021. Her general interests include data mining area, big data field, wireless sensor networks, fuzzy systems, machine learning, deep learning, and the Internet Things.

Hamid Garmani received the Ph.D. degrees from University Sultan Moulay Slimane, Morocco, in 2020. His research interests include network economics, network security, applications of game theory in wireless networks, and radio resource management.

Junaid Qadir is currently pursuing the Ph.D. degree with the Department of Electrical, Electronic and Telecommunications Engineering, and Naval Architecture (DITEN), University of Genova, Italy. He is also a Research Collaborator with the Department of Signal Theory, Communications and Telematics Engineering, University of Valladolid, Spain. He has published many research articles in highly reputed international journals and conferences.

Abdellatif Hair currently works as a Full Professor with the Department of Computer, FST Beni Mellal, Morocco, and a member of the LAMSC Laboratory. His research interests include object-oriented analysis/design, security of mobile agents, wireless sensor network (WSN), data warehousing, and machine learning (ML).