Information Processing and Management 57 (2020) 102122

Contents lists available at ScienceDirect

Information Processing and Management

journal homepage: www.elsevier.com/locate/infoproman

Towards a real-time processing framework based on improved distributed recurrent neural network variants with fastText for social big data analytics

Badr Ait Hammou[*,a], Ayoub Ait Lahcen[a,b], Salma Mouline[a]

[a] LRIT, Associated Unit to CNRST (URAC 29), Rabat IT Center, Faculty of Sciences, Mohammed V University, Rabat, Morocco
[b] LGS, National School of Applied Sciences (ENSA), Ibn Tofail University, Kenitra, Morocco

ARTICLE INFO

Keywords:
Big data
FastText
Recurrent neural networks
LSTM
BiLSTM
GRU
Natural language processing
Sentiment analysis
Social big data analytics

ABSTRACT

Big data generated by social media stands for a valuable source of information, which offers an excellent opportunity to mine valuable insights. Particularly, User-generated contents such as reviews, recommendations, and users' behavior data are useful for supporting several marketing activities of many companies. Knowing what users are saying about the products they bought or the services they used through reviews in social media represents a key factor for making decisions. Sentiment analysis is one of the fundamental tasks in Natural Language Processing. Although deep learning for sentiment analysis has achieved great success and allowed several firms to analyze and extract relevant information from their textual data, but as the volume of data grows, a model that runs in a traditional environment cannot be effective, which implies the importance of efficient distributed deep learning models for social Big Data analytics. Besides, it is known that social media analysis is a complex process, which involves a set of complex tasks. Therefore, it is important to address the challenges and issues of social big data analytics and enhance the performance of deep learning techniques in terms of classification accuracy to obtain better decisions.

In this paper, we propose an approach for sentiment analysis, which is devoted to adopting fastText with Recurrent neural network variants to represent textual data efficiently. Then, it employs the new representations to perform the classification task. Its main objective is to enhance the performance of well-known Recurrent Neural Network (RNN) variants in terms of classification accuracy and handle large scale data. In addition, we propose a distributed intelligent system for real-time social big data analytics. It is designed to ingest, store, process, index, and visualize the huge amount of information in real-time. The proposed system adopts distributed machine learning with our proposed method for enhancing decision-making processes. Extensive experiments conducted on two benchmark data sets demonstrate that our proposal for sentiment analysis outperforms well-known distributed recurrent neural network variants (i.e., Long Short-Term Memory (LSTM), Bidirectional Long Short-Term Memory (BiLSTM), and Gated Recurrent Unit (GRU)). Specifically, we tested the efficiency of our approach using the three different deep learning models. The results show that our proposed approach is able to enhance the performance of the three models. The current work can provide several benefits for researchers and practitioners who want to collect, handle, analyze and visualize several sources of information in real-time. Also, it can contribute to a better understanding of public opinion and user behaviors using our proposed system with the improved

* Corresponding author.
E-mail addresses: badr.aithamou@gmail.com (B. Ait Hammou), ayoub.aitlahcen@univ-ibntofail.ac.ma (A. Ait Lahcen), salma.mouline@um5.ac.ma (S. Mouline).

https://doi.org/10.1016/j.ipm.2019.102122
Received 25 May 2019; Received in revised form 12 June 2019; Accepted 7 September 2019

variants of the most powerful distributed deep learning and machine learning algorithms. Furthermore, it is able to increase the classification accuracy of several existing works based on RNN models for sentiment analysis.

1. Introduction

In the era of Big Data, the world's largest technology organizations like Microsoft, Amazon, and Google have collected massive amounts of data estimated at the size of exabytes or larger. Social media companies like Twitter, Facebook, and YouTube have billions of users. Therefore, many businesses are employing social media to keep in touch with their clients, and promote the services and products offered. Clients also adopt social media to get information about interesting services or goods (Najafabadi et al., 2015; Salehan & Kim, 2016; Xiang, Schwartz, Gerdes Jr, & Uysal, 2015).

The tremendous growth of social media with the users' generated data provide an excellent opportunity to mine valuable insights and understand better users' behaviors. This has motivated the development of big data solutions to solve many real-life issues (Najafabadi et al., 2015; Salehan & Kim, 2016; Xiang et al., 2015).

Generally, the term Big Data stands for data sets whose volume exceeds the capabilities of conventional tools to capture, manage, analyze and store data effectively (Bello-Orgaz, Jung, & Camacho, 2016; García-Gil, Luengo, García, & Herrera, 2019). The concept of Big data is characterized by the 5Vs (i.e., volume, velocity, variety, veracity, value).

- Volume: This characteristic refers to massive volumes of data generated every second. Finding valuable insights through the exploration and analysis processes create serious problems for traditional tools. For instance, Flickr generates nearly 3.6 TB of data and Google handles approximately 20,000 TB each day (Zhang, Yang, Chen, & Li, 2018). Furthermore, as stated by the authors (McAfee, Brynjolfsson, Davenport, Patil, & Barton, 2012), in 2012, approximately 2.5 exabytes of data were generated each day. This amount of data doubles every 40 months. According to (García-Gil et al., 2019; García-Gil, Ramírez-Gallego, García, & Herrera, 2017), it is estimated that by 2020, the digital world will reach 44 zettabytes (i.e., 44 trillion gigabytes), which is 10 times larger than 4.4 zettabytes in 2013.
- Velocity: The speed at which data are generated and should be processed. In particular, with the proliferation of digital devices such as sensors and smartphones, the data generated has witnessed an unprecedented rate of data growth, which poses serious challenges to handle streaming data and perform real-time analytics (Bello-Orgaz et al., 2016; Gandomi & Haider, 2015).
- Variety: It indicates the various types of data that may be available in a structured, semi-structured, or unstructured format. Structured data constitutes the smallest percentage of all the existing data, where relational databases represent a typical example of structured data. Semi-structured refers to data that does not conform to strict standards like XML (Extensible Markup Language). The third type is unstructured data, which represents more than 75% of big data. It includes audio, text, video and images (Bello-Orgaz et al., 2016; Chen, Mao, & Liu, 2014; Gandomi & Haider, 2015; Zhang et al., 2018).
- Veracity: This characteristic was coined by IBM as the fourth V. It stands for the correctness and accuracy of data. In particular, with many forms of big data, quality and correctness are less controllable. For instance, the sentiments of users in social media are inherently uncertain because they involve human judgment. Nevertheless, they contain valuable information. Thus, the necessity to deal with uncertain and imprecise data is another aspect of Big Data, which should be addressed using the appropriate tools (Bello-Orgaz et al., 2016; Gandomi & Haider, 2015; García-Gil et al., 2019).
- Value: Oracle added the so-called value as the fifth characteristic of Big data. At a simplistic level, big data has no intrinsic value. It becomes valuable only when we are able to derive the insights required to meet a particular need or address a problem. In other words, value is a key feature for any big data application as it allows generating useful business information (Bello-Orgaz et al., 2016; Lee, 2017).

In the era of Big data, several companies around the world are using various solutions and techniques to analyze their huge amount of data in order to get meaningful insights. These techniques are called Big Data Analytics. In particular, the term big data analytics encompasses several algorithms, advanced statistics and applied analytics, which are used for various purposes such as prediction, classification, decision-making, and so on (Jimenez-Marquez, Gonzalez-Carrasco, Lopez-Cuadrado, & Ruiz-Mezcua, 2019; Najafabadi et al., 2015; Saggi & Jain, 2018; Wang, Kung, & Byrd, 2018a).

More particularly, in addition to analyzing a large amount of data, Big Data Analytics creates serious challenges for machine learning techniques and data analysis task, including noisy data, highly distributed input data sources, high-dimensionality, limited labeled data, and so on. Furthermore, there are other real problems in Big Data Analytics such as data indexing, data storage, and information retrieval. As a result, more sophisticated data analysis and data management tools are necessary to handle the massive amount of data and deal with various real-world problems in the context of Big Data (Jimenez-Marquez et al., 2019; Najafabadi et al., 2015).

Due to the rapid development of social media, the huge amount of reviews generated caught the attention of several organizations, governments, businesses, and politicians to know the public opinion on various issues and understand the users' behaviors for specific purposes. Moreover, many users follow the opinions and feedback of others to decide the quality of a product or service before purchase. Therefore, sentiment analysis plays an essential role in decision-making tasks. It represents a practical technique, which is commonly used to deduce the sentiment polarity from people's opinions (Del Vecchio, Mele, Ndou, & Secundo, 2018;

Jianqiang & Xiaolin, 2017; Ngai, Tao, & Moon, 2015; Pham & Le, 2018; Ragini, Anand, & Bhaskar, 2018; Rezaeinia, Rahmani, Ghodsi, & Veisi, 2019; Stieglitz, Mirbabaie, Ross, & Neuberger, 2018; Valdivia, Luzón, & Herrera, 2017).

Deep learning (DL) stands for an extremely active research area in the machine learning (ML) community. It encompasses a set of learning algorithms, which are intended to automatically learn the hierarchical representations and extract the high-level features based on deep architectures. In particular, deep learning models have achieved remarkable success in various natural language processing tasks, including text classification and sentiment analysis (Chen et al., 2014; Rezaeinia et al., 2019; Zhang, Yang, & Chen, 2016).

Due to the different characteristics of big data, the design of an efficient big data system based on deep learning requires the consideration of many issues. In particular, because of the volume and the variety of big data sources, it is tricky to integrate effectively data collected from various distributed data sources. For example, more than 175 million tweets (i.e., unstructured data, which include images, videos, text, and so on) are posted by millions of user accounts in the whole world. In addition, it is necessary to store and handle the collected heterogeneous data efficiently. For instance, Facebook needs to manage, store and analyze more than 30 petabytes of data. Moreover, in order to take advantage of big data analytics, there is a need to analyze big data based on real-time, near real-time or batch processing. As a result, enhancing the performance of techniques used for a variety of tasks like classification, prediction, and visualization, is crucial for improving decision-making processes. Thus, many companies can benefit from the advantages of artificial intelligence with big data, increasing revenue by strengthening customer relationships (Hu, Wen, Chua, & Li, 2014; Jimenez-Marquez et al., 2019).

In this work, instead of directly applying DL models to real-world problems, one promising achievement is to improve the performance of the most powerful deep leaning models for social big data analytics. In general, the concept of social big data analytics can be deemed as the intersection between big data analytics and social media. Its main objective is to take advantage of the efforts of the two fields for analyzing and extracting relevant knowledge from the huge amount of social media data. However, real-time social big data analytics aims to manage the complexity of conducting social big data analytics in real-time, to process unbounded data streams, which is the goal of our proposed framework.

The contributions of this paper are summarized as follows:

- Instead of using deep learning models to classify the reviews directly, we proposed a procedure based on fastText word embedding and Recurrent neural network variants for sentiment analysis, which is devoted to representing textual data efficiently.
- We designed an efficient strategy based on machine learning, which is tailored to improve the performance in terms of classification accuracy of three distributed Recurrent neural network variants, namely, Distributed Long Short-Term Memory (LSTM), Distributed Bidirectional Long Short-Term Memory (BiLSTM) and Distributed Gated Recurrent Unit (GRU) models.
- We proposed a distributed intelligent system for real-time social big data analytics. It is based on a set of steps, which is dedicated not only to ingest, store, process, index, and visualize a large amount of information in real-time, but also it adopts a set of distributed machine learning and deep learning techniques for effective classification, prediction, and real-time analysis of customer behavior and public opinion.
- We conducted a set of experimentations using two real-world data sets. The experimental results demonstrate that our proposal yields better classification accuracy than existing state-of-the-art methods. Moreover, it is able to improve the performance of several existing works.

The rest of the paper is structured as follows. Next section outlines the related works. Section 3 describes Recurrent neural network (RNN) variants and fastText for sentiment analysis. Section 4 details our proposal. Section 5 describes the experiments. Finally, Section 6 concludes this paper.

2. Related work

With the advent of social networks, big data analytics played an important role in decision-making processes. Due to the remarkable success of Deep Learning (DL) approaches, various solutions have been proposed to cope with the challenges related to Natural language processing tasks.

In particular, several deep learning models for natural language processing have been designed based on employing word vector representations. For instance, Mikolov, Sutskever, Chen, Corrado, and Dean (2013) proposed the Word2Vec algorithm, a deep learning model for vector representations of the words, which is intended to convert words into meaningful vectors. Pennington, Socher, and Manning (2014) put forward a model for learning vector representations of words, namely, Global Vectors for Word Representation (GloVe). It is based on the global matrix factorization and local context window methods. Yu, Wang, Lai, and Zhang (2017) designed a word vector refinement model, which is intended to refine existing pre-trained word vectors such as Word2vec and GloVe. The key idea of the model is adjusting the word representations to capture both semantic and sentiment word vectors. Rezaeinia et al. (2019) developed an approach called Improved Word Vector (IWV), which is tailored to improve the accuracy of two pre-trained word embeddings, Word2Vec and GloVe vectors for sentiment analysis task. In particular, it employs four different approaches, POS tagging, Word2Vec with GloVe, word position and lexicon-based approach. Bojanowski, Grave, Joulin, and Mikolov (2017) designed an approach for learning word representations by taking into account subword information. The key idea is to incorporate character n-grams into the skip-gram model, where each word is considered as a bag of character n-grams. Each character is represented as a vector representation.

However, other researchers have tended to focus on the problem of sentiment analysis and text classification based on word embedding with deep learning. Kim (2014) proposed three deep learning models, which are designed based on Convolution Neural Network (CNN) and pre-trained word2vec vectors for sentence classification. Kalchbrenner, Grefenstette, and Blunsom (2014) proposed a deep learning architecture for the semantic modeling of sentences, called dynamic convolutional neural network (DCNN). It is devoted to analyzing and representing the semantic content of sentences for classification tasks. Chen, Xu, He, and Wang (2017) designed an approach for sentence-level sentiment classification. It is based on two steps. First, it classifies sentences into different types. Second, each group of sentences is fed into a convolutional neural network model for performing sentiment analysis task. Jianqiang, Xiaolin, and Xuejun (2018) introduced an approach, which aims to combine glove word embeddings with n-grams features and the polarity score features to represent the features of tweets. Then, a deep CNN model is adopted to process these features and perform the sentiment classification task. While, Alharbi and de Doncker (2019) developed a convolutional neural network model, which is designed to incorporate user behavioral information according to each tweet.

On the other hand, Iyyer, Manjunatha, Boyd-Graber, and Daumé III (2015) proposed a deep neural network for text classification, namely, deep averaging network (DAN). It is tailored to process an unweighted average of word vectors through multiple hidden layers before performing the classification task. Socher, Pennington, Huang, Ng, and Manning (2011) introduced a framework based on semi-supervised recursive autoencoders for predict sentence-level sentiment distributions. Socher et al. (2013) developed an approach called Recursive Neural Tensor Network (RNTN). The intuition behind this model is that it calculates the compositional vector representations for phrases. Then, the representations will be utilized as features for classifying each phrase.

The advent of sequence modeling has inspired many researchers to design effective approaches for sentiment analysis. For example, Wang, Liu, Chengjie, Wang, and Wang (2015) proposed Long Short-Term Memory (LSTM) recurrent network for twitter sentiment prediction. It aims to compose word representations through a flexible compositional function. Tai, Socher, and Manning (2015) introduced an approach called Tree-LSTM. It represents a generalization of the conventional LSTM architecture to tree-structured network topology. Wang, Yu, Lai, and Zhang (2016) developed a method called regional CNN-LSTM model for dimensional sentiment analysis. It is designed based on regional CNN and LSTM. In particular, unlike a conventional CNN which process the whole text as input, the regional CNN considers an input text as a set of regions (i.e., sentences), therefore, it treats each sentence separately. Then, LSTM model is adopted to consider long-distance dependency across sentences in the prediction process. Vosoughi, Vijayaraghavan, and Roy (2016) proposed an approach for learning tweet embeddings. called Tweet2Vec. It is designed based on character-level CNN-LSTM encoder-decoder. Tien and Le (2017) introduced an approach based on CNN and LSTM, which aims to synthesize feature vectors for performing sentiment analysis. Liu and Guo (2019) developed an approach, called attention-based bidirectional long short-term memory with convolution layer (AC-BiLSTM) for sentiment analysis. This approach is based on a bidirectional LSTM, attention mechanism and the convolutional layer. Specifically, the convolutional layer is used to extract higher-level phrase representations. BiLSTM is adopted to access the preceding and succeeding context representations. While the attention mechanism is intended to give focus on the information produced by the hidden layers of BiLSTM.

3. Recurrent neural network variants with fastText for sentiment analysis

This section presents the word embedding technique called fastText, with Recurrent neural network, Long short-term memory, Bidirectional long short-term memory and Gated recurrent unit methods for Sentiment analysis.

3.1. FastText: word representation learning

With the success of deep learning, the models based on neural networks have become increasingly popular to convert words into meaningful vectors. These continuous representations of words stand for the fundamental building blocks for many Natural Language Processing (NLP) applications, such as clustering, information retrieval and text classification (Mikolov, Grave, Bojanowski, Puhrsch, & Joulin, 2018; Rezaeinia et al., 2019).

The fastText is one of the most popular word embedding techniques based on neural networks, which is created by Bojanowski et al. at Facebook's AI Research lab (Mikolov et al., 2018). It is tailored to learn high-quality representations of words while considering the morphology.

In general, fastText is based on the continuous skipgram model introduced by Mikolov et al. (2013) and subword model.

3.1.1. Skipgram model with negative sampling

Let W be the size of the word vocabulary, where each word is identified by its index denoted $w \in \{1, \cdots, W\}$. Given a training corpus defined as a sequence of words $w_1, \cdots, w_T$, the main goal of the skipgram algorithm is to maximize the following objective function:

$$O = \sum_{t=1}^{T} \sum_{c \in C_t} \log(p(w_c | w_t)) \tag{1}$$

where C_t represents the context, which is defined as the set of indices of words surrounding the word w_t.

The problem of predicting context words can be considered as a set of independent binary classification tasks. The objective is to predict the presence or absence of context words. In particular, for the word at position t, all context words are considered as positive examples, and negatives are sampled randomly from the vocabulary. As a result, the objective function can be defined as follows:

$$\log\left(1 + e^{-s(w_t, w_c)}\right) + \sum_{n \in N_{t,c}} \log\left(1 + e^{s(w_t, n)}\right) \tag{2}$$

where $N_{t,c}$ stands for a set of negative examples sampled from the vocabulary.

Let u_{w_t} and v_{w_c}, two vectors corresponding to the words w_t and w_c, respectively. The function $s(w_t, w_c)$ can be computed as the scalar product between word and context vectors as follows:

$$s(w_t, w_c) = u_{w_t}^T v_{w_c} \tag{3}$$

3.1.2. Subword model

In this step, each word is considered as a bag of character n-grams. Let D be a dictionary of n-grams of size $|D|$. For each word w, D_w denotes the set of n-grams appearing in w, where each vector representation z_d is associated to each n-gram denoted d. Thus, each word w is represented by the sum of the vector representations of its n-grams.

$$s(w, c) = \sum_{d \in D_w} z_d^T v_c \tag{4}$$

This model facilitate sharing the representations across words, which allows to learn reliable representation for rare words (Bojanowski et al., 2017).

3.2. Deep learning architectures for sequence modeling

3.2.1. Recurrent neural network

Recurrent neural network (RNN) is a class of neural network architectures which is devoted for modeling sequential data. In particular, it is extremely useful for several natural language processing tasks such as sentiment analysis, text classification, speech-to-text, machine translation, and so on.

Let $V = (v_1, \cdots, v_N)$ be an input sequence, which represents a review composed of N words as a set of N fastText vector representations. Each word w_i is represented by a $D-$dimensional vector $v_i \in \mathbb{R}^D$.

The RNN model iterates over the N words to calculate the hidden states represented as $(h_1, \cdots, h_N)$, and produces the output. More precisely, at each time step t, the RNN model takes both the word embedding vector $v_t \in \mathbb{R}^D$ and the previous hidden state $h_{t-1} \in \mathbb{R}^M$ as input. It calculates the current hidden state $h_t \in \mathbb{R}^M$ based on the following equation:

$$h_t = H(W_h v_t + U_h h_{t-1} + b_h) \tag{5}$$

where H denotes a non-linear activation function such as sigmoid, hyperbolic tangent function. The symbols $W_h \in \mathbb{R}^{M \times D}$ and $U_h \in \mathbb{R}^{M \times M}$ refer to the weight matrices, $b_h \in \mathbb{R}^M$ represents the bias term, M denotes the number of hidden units (Graves, Mohamed, & Hinton, 2013; Kim, Jernite, Sontag, & Rush, 2016; Sutskever, Martens, & Hinton, 2011; Tai et al., 2015).

Next, based on the hidden state h_t, the predicted output y_t is defined as follows:

$$y_t = softmax(W_y h_t + b_y) \tag{6}$$

where $W_y \in \mathbb{R}^{C \times M}$ is the weight matrix, C is the number of output classes, b_y refers to the bias term.

In general, for training the RNN model, an extension of the backpropagation algorithm, known as Backpropagation Through Time (BPTT), is adopted to perform this task. However, in practical situations, the recurrent neural network model faces the vanishing/exploding gradients, a real problem that negatively affects the RNN performance, which prevents it from learning long-term dependencies (Kim et al., 2016; Mohammadi, Al-Fuqaha, Sorour, & Guizani, 2018).

3.2.2. Long short-term memory

Long Short-Term Memory network (LSTM) stands for a special type of recurrent neural network architecture, which has shown great promise in several natural language processing tasks. It is mainly tailored to learn long-term dependencies by adopting memory cells to store information. In addition, a system of gating units is employed to control the information flow into and out of the cell. As a result, LSTM architecture is able to overcome the vanishing gradient, and work better than the standard recurrent neural network (Graves & Schmidhuber, 2005; Greff, Srivastava, Koutník, Steunebrink, & Schmidhuber, 2017; Hochreiter & Schmidhuber, 1997; Kim et al., 2016; Mohammadi et al., 2018; Song, Park, & Shin, 2019).

Let N be the number of LSTM cells. One LSTM cell has a hidden state h_t, a memory cell c_t, and three different gates, namely, input gate i_t, forget gate f_t, and an output gate o_t.

- **Forget gate**: it is devoted to controlling how much of the previous long-term state is read into the current cell state.
- **Input Gate**: it controls how much of the current input is added into the cell state.
- **Output Gate**: it decides which parts of the long term state should be in the next hidden state.

Each one of the gates adopts the sigmoid as an activation function, and produces a value in the interval [0, 1].

$$\begin{aligned} i_t &= \sigma(W_i v_t + U_i h_{t-1} + b_i) \\ f_t &= \sigma(W_f v_t + U_f h_{t-1} + b_f) \\ o_t &= \sigma(W_o v_t + U_o h_{t-1} + b_o) \end{aligned} \tag{7}$$

where $\sigma(.)$ represents the sigmoid function. W_i, W_f, W_o, U_i, U_f, U_o are the weight matrices for the gates. b_i, b_f, b_o are the bias weights.

Generally, at the time step t, the LSTM architecture takes as input the word representation vector $v_t \in \mathbb{R}^D$, the previous hidden state $h_{t-1} \in \mathbb{R}^M$ and the previous memory cell vector c_{t-1}. It calculates h_t, c_t as follows:

$$\begin{aligned} g_t &= \tan h(W_g v_t + U_g h_{t-1} + b_g) \\ c_t &= f_t \odot c_{t-1} + i_t \odot g_t \\ h_t &= o_t \odot \tan h(c_t) \end{aligned} \tag{8}$$

where the notation $\odot$ denotes the element-wise multiplication, *tanh*(.) refers to the hyperbolic tangent function, W_g and U_g are the weight matrices, b_g is the bias term.

3.2.3. Bidirectional long short-term memory

Bidirectional Long Short-Term Memory (BiLSTM) represents an extension of the unidirectional LSTM. Specifically, it employs two LSTMs. The first one is the forward LSTM denoted by $\overrightarrow{LSTM}$. It processes the input sequence from left to right. The second is the backward LSTM, which is denoted by $\overleftarrow{LSTM}$. It processes the sequence in reverse order, from right to left (Chen et al., 2017; Graves et al., 2013; Lin, Xu, Luo, & Zhu, 2017; Tai et al., 2015).

The key idea behind the Bidirectional LSTM model is to exploit both future and past context. At each time step t, it generates the hidden state by concatenating the forward hidden state $\overrightarrow{h_t}$ and the backward hidden state $\overleftarrow{h_t}$ defined as follows:

$$\begin{aligned} \overrightarrow{h_t} &= \overrightarrow{LSTM}(v_t, \overrightarrow{h_{t-1}}) \\ \overleftarrow{h_t} &= \overleftarrow{LSTM}(v_t, \overleftarrow{h_{t+1}}) \end{aligned} \tag{9}$$

where the forward hidden state $\overrightarrow{h_t}$ is calculated based on the previous hidden state $\overrightarrow{h_{t-1}}$ and the input v_t, while the backward hidden state $\overleftarrow{h_t}$ is computed based on the future hidden state $\overleftarrow{h_{t+1}}$ and v_t.

3.2.4. Gated recurrent unit

Gated recurrent unit network stands for a variant of the standard LSTM network. It designed to simplify the complex architecture of LSTM networks by reducing the number of trainable parameters in each cell. It has proven its ability to provide better performance than other existing models.

As a traditional LSTM unit, Gated Recurrent Unit (GRU) adopts a gating mechanism to control information flow inside the unit. It has only two gates for controling the hidden state, known as the reset gate r_t and the update gate u_t (Cho et al., 2014; Chung, Gulcehre, Cho, & Bengio, 2014).

Mathematically, these two gates are defined as follows:

$$\begin{aligned} r_t &= \sigma(W_r v_t + U_r h_{t-1} + b_r) \\ u_t &= \sigma(W_u v_t + U_u h_{t-1} + b_u) \end{aligned} \tag{10}$$

At the time step t, the activation h_t and the candidate activation $\hat{h}_t$ of GRU are computed as follows:

$$\begin{aligned} \hat{h}_t &= \tan h(W_{\hat{h}} v_t + U_{\hat{h}}(r_t \odot h_{t-1}) + b_{\hat{h}}) \\ h_t &= u_t \hat{h}_t + (1 - u_t) \odot h_{t-1} \end{aligned} \tag{11}$$

where W_r, W_u, $W_{\hat{h}}$, U_r, U_u, $U_{\hat{h}}$ are the weight matrices. b_r, b_h, $b_{\hat{h}}$ are the bias terms.

4. Our proposal

In this section, we present a distributed intelligent system for social big data analytics, and our proposal for improving distributed RNN variants. The key idea is to perform real-time analysis, improve the classification performance of the most powerful RNN models, and handle large-scale data sets using parallel and distributed training.

4.1. Proposed distributed intelligent architecture for real-time big data analytics

To deal with Big Data issues, we have proposed a distributed intelligent system for real-time stream processing, which is based on a set of tools devoted to handling the large amount of data and perform big data analytics in an efficient and effective way.

The system combines multiple layers into a single architecture. This pipeline architecture aims to run a sequence of steps needed for processing and analyzing data. As illustrated in Fig. 1, the system consists of a set of stages organized as follows:

1. Big Data acquisition
2. Big Data storage layer
3. Big Data ingestion layer
4. Big Data processing layer
5. Distributed deep learning for natural language processing layer
6. Distributed machine learning layer
7. Big Data indexing layer
8. Real-time visualization layer

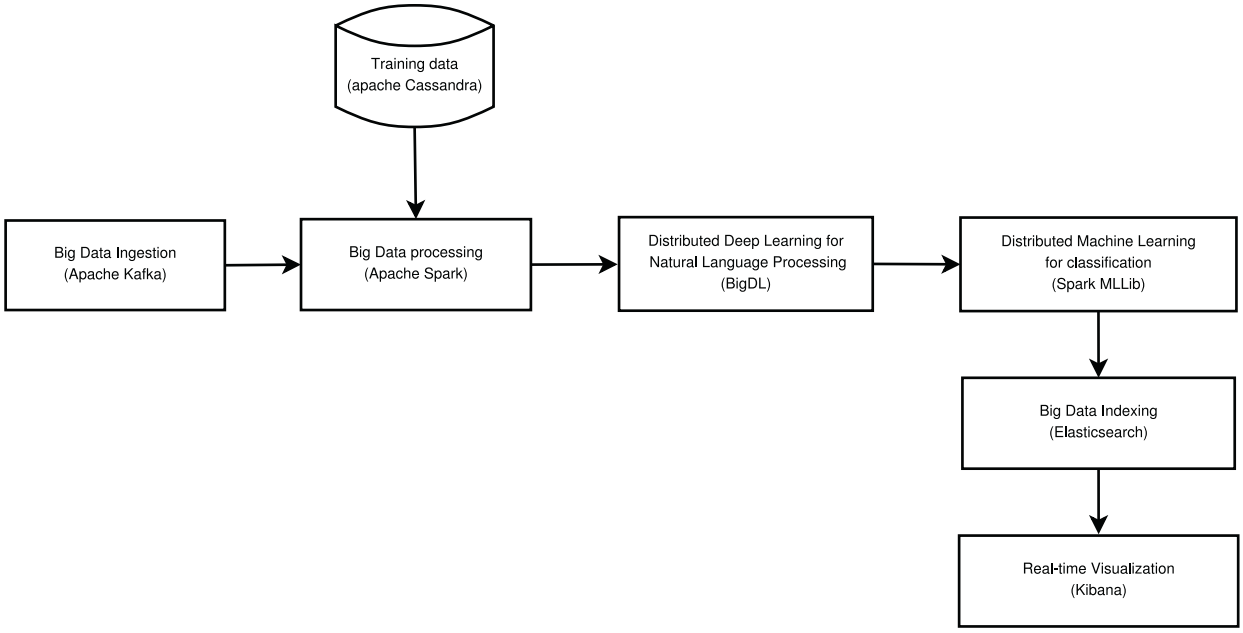

Fig. 1. Flowchart of the distributed real-time system for social Big Data Analytics.

4.1.1. Big data acquisition

This layer is devoted to collecting data from several internal and external sources. The data can be available as structured data, semi-structured data, and unstructured data. In this work, as a case study, the data to be processed is textual data from several social media sources, i.e., users' reviews representing the opinions of users on several topics, to provide the insights required for achieving specific objectives, such as solving serious business problems, real-time analysis of customer behavior and public opinion.

4.1.2. Big data ingestion layer

In the big data era, real-time information is generated continuously. To deliver the data to several types of receivers, an effective way is needed to perform this task. This layer is responsible to quickly deliver information collected from various data sources for facilitating the consumption of millions of messages in the pipeline architecture. Therefore, we have adopted Apache Kafka, a distributed streaming platform, which was developed at LinkedIn. It is a fault-tolerant, scalable, real-time publish-subscribe messaging system.

In general, Kafka is based on a set of concepts such as topic, producer, consumer, and broker. A topic represents a stream of records of a particular type. A producer enables an application to publish a stream of messages to one or many topics. All the published records are stored at a set of servers called brokers. Whereas a consumer enables an application to subscribe to at least one or many topics and handle the stream of messages produced (Jan et al., 2019; Kafka, 2019; Kreps, Narkhede, Rao et al., 2011).

4.1.3. Big data storage layer

This layer is intended to store the data needed for training and testing the distributed deep learning models. As NoSQL represents a category of databases tailored for handling Big Data. We have adopted Apache Cassandra, a distributed NoSQL database management system, originally developed by Facebook. It is devoted to managing large amounts of data spread out across many nodes, possibly different data centers. It offers several advantages including scalability, high availability, fault-tolerance without compromising performance (Cassandra, 2019; Lakshman & Malik, 2010).

4.1.4. Big data processing layer

The main objective of this layer is to process the collected data from the previous layer, perform the pre-processing tasks such as data cleaning, tokenization, and converting words into the corresponding vector representations.

This layer relies on Apache Spark for performing large-scale data processing as it is a fast, an in-memory cluster computing framework. It has emerged as a strong successor to Apache Hadoop, essentially due to its higher capabilities for richer and faster analysis. Spark can run programs up 10x faster than Apache Hadoop on disk, or 100x faster in-memory (Ait Hammou, Ait Lahcen, & Mouline, 2018; Spark, 2019).

In particular, Apache Spark is based on the so-called Resilient Distributed Dataset (RDD), which stands for Spark's fundamental abstraction. It is a read-only collection of partitioned data, which allows effective data reuse in a wide range of applications. Generally, RDD is a fault-tolerant data structure. It presents several advantages, such as optimizing distributed data placement, and the manipulation by employing a set of operators (Ait Hammou et al., 2018; Spark, 2019; Zaharia et al., 2012; Zaharia, Chowdhury, Franklin, Shenker, & Stoica, 2010).

4.1.5. Distributed deep learning for natural language processing layer

This layer is devoted to training distributed deep learning models on large amounts of data spread out across several nodes. In particular, we have adopted three distributed Recurrent neural network variants for natural language processing, namely, Distributed Long Short-Term Memory (LSTM), Distributed Bidirectional Long Short-Term Memory (BiLSTM) and Distributed Gated Recurrent Unit (GRU).

In order to improve the performance of the aforementioned models in terms of classification accuracy, this step aims to learn the appropriate representation with respect to each review in the data set. We have employed BigDL framework, a deep learning library designed mainly for training large-scale distributed deep learning. It is developed by Intel for Apache Spark. BigDL enables writing distributed deep learning for big data applications as standard Apache Spark programs and running on a cluster of servers (BigDL, 2019; Wang et al., 2018b).

4.1.6. Distributed machine learning layer

This step is dedicated to classify textual data, by exploiting the optimal representations generated by our proposal. In order to perform this task in the context of Big data, we have chosen a Spark's machine learning library called MLlib, which aims to make the development of large-scale machine learning applications scalable and easy to use. The MLlib provides a variety of fast and scalable implementations of machine learning algorithms for various tasks, such as dimensionality reduction, regression, clustering, and classification. It supports several languages, including Java, Python, Scala and R (Meng et al., 2016; MLlib, 2019)

4.1.7. Big data indexing layer

This layer is intended to index data for providing efficient data storage, real-time search, and analyzing large amounts of data quickly. Therefore, in order to power our pipeline architecture by these functionalities, Elasticsearch is utilized as the underlying engine at this stage, which is a highly scalable framework built on top of Apache Lucene, a full-text search-engine library. It is a distributed search and analytics engine, where all fields in each stored document are indexed and searchable. Elasticsearch is primarily dedicated to supporting not only the search functionality but also complex aggregations (Elasticsearch, 2019; Gormley & Tong, 2015).

4.1.8. Real-time visualization layer

In order to explore and understand efficiently the data indexed in the previous stage, the current step is dedicated to real-time big data visualization. Particularly, it aims to perform big data analysis in a more comprehensible way, for gaining useful insights and extracting relevant information needed for serving specific objectives of several companies, like, business intelligence, real-time analysis of customer behavior and public opinion, and so on. Therefore, this step is based on a powerful platform called Kibana, which is devoted to visualizing data stored in Elasticsearch indexes. It exploits the powerful search and indexing functionalities of Elasticsearch framework available via its RESTful API to display several graphics. Kibana provides flexible analytics and visualization to facilitate the task of understanding large volumes of data (Kibana, 2019).

4.2. Proposed method for sentiment analysis

This subsection depicts our proposed technique for improving the performance of RNN variants. Fig. 2 illustrates the different steps.

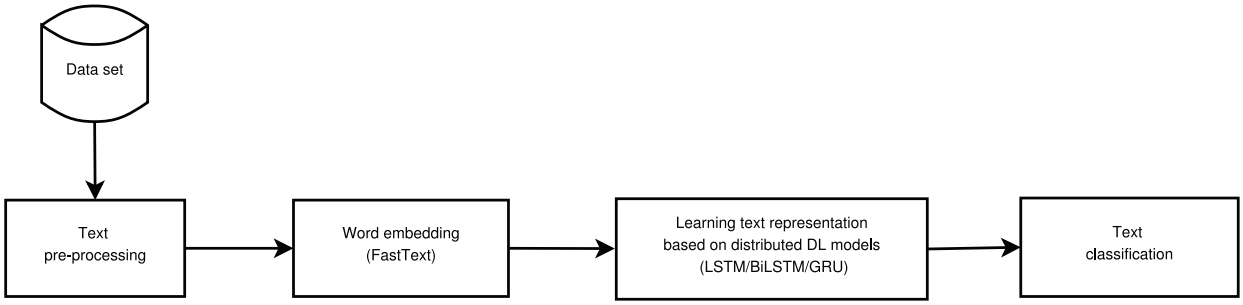

Fig. 2. Flowchart of the proposed method.

4.2.1. Text pre-processing

Text pre-processing is an essential stage for text analytics. It refers to the process of preparing the data for sentiment analysis. As the reviews are usually composed of noisy and poorly structured sentences, it is necessary to transform the content of each review into the appropriate format to reduce the negative impact of the noise and facilitate text analytics. Therefore, this step is intended to perform a series of pre-processing such as removing the numbers, URLs, hashtags, and so on.

4.2.2. Embedding layer

In this step, the key idea of the embedding layer is to learn representations of words. Let $R^{(i)} = (w_1, \cdots, w_N)$ be an input sequence, which represents a review composed of N words after padding. In order to process the review $R^{(i)}$ based on a recurrent neural network variant model, it is common to transform textual data into numerical vectors, where each vector capture hidden information about the language. Based on the pre-trained fastText word representations (FastText, 2019), each review $R^{(i)}$ in the data set can be converted into a sequence of vectors denoted as follows:

$$V^{(i)} = FastText(R^{(i)}) = (v_1, \cdots, v_N) \tag{12}$$

where each word w_i is represented by a D−dimensional vector $v_i \in \mathbb{R}^D$.

4.2.3. Learning text representations based on distributed RNN variants

The main goal of this step is to learn the optimal feature representations of each input sequence based on a more sophisticated procedure.

Let $T = \{(x_i,\ y_i),\ i = 1,\ \cdots, |T|\}$ be the training data set, which is composed of $|T|$ instances. Each x_i and y_i stand for the sequence of word vectors and the label corresponding to a review $R^{(i)}$, respectively. Also, let k be the parameter of representation, which determines the number of functions needed to infer the optimal representation. The first purpose is to take each input sequence of vectors $x_i = V^{(i)}$, and convert it into an optimal vector representation $O^{(i)} \in \mathbb{R}^k$ according to the following equation:

$$O^{(i)} = g(V^{(i)}) = (o_1^{(i)}, \cdots, o_k^{(i)}) \tag{13}$$

The key idea behind the function $g(.)$ is to consider the problem of learning representation of a sequence as a set of k sub-problems, where each sub-problem can be resolved independently by minimizing a separate objective function. The element $o_1^{(i)} \in \mathbb{R}$ denotes the value estimated after the minimization process. The set of functions F is typically expressed as follows:

$$\begin{aligned} F &= (F_j,\ j = 1, \cdots, k) \\ F_j &= -\frac{1}{|T|} \sum_{i=1}^{|T|} \sum_{c=1}^{C} y_i^{(c)} \log(\hat{p}_i^{(c)}) \end{aligned} \tag{14}$$

where $y_i^{(c)}$ equals 1 if the ground-truth label for the instance x_i is c, else, it is equal to 0. C is the number of output labels, and $\hat{p}_i^{(c)}$ denotes the estimated probability for the label c. Each function F_j is a cross entropy loss function associated to a deep learning model, which is able to handle variable-length input sequences and learn long-term dependencies in sequential data.

Therefore, each classification problem can be solved based on a distributed Recurrent neural network variant. Specifically, in this step, we have adopted three models, namely, Long short-term memory (LSTM), Bidirectional long short-term memory (BiLSTM) and Gated recurrent unit (GRU). Each model stacks multiple layers as defined below:

- **Input Layer**: This first layer receives each review $R^{(i)}$ as a sequence of 300-dimensional vectors $V^{(i)}$.
- **Distributed Recurrent neural network variant Layer**: This second layer employs a LSTM, BiLSTM or GRU model to process the input sequences and output the hidden state at the last time step denoted $h^{(i)}$.
- **Dense Layer**: The third one is a fully connected layer of 100 nodes, which processes the last vector state $h^{(i)}$, and produces a hidden representation $dl^{(i)} \in \mathbb{R}^{100}$. Leaky ReLU is used as the nonlinear activation function with the parameter $\alpha = 0.3$ (Maas, Hannun, & Ng, 2013). Dropout (Hinton, Srivastava, Krizhevsky, Sutskever, & Salakhutdinov, 2012) is used to prevent the overfitting.

- **Softmax Layer**: The softmax layer estimates the probability distribution vector $\hat{p}^{(i)} \in \mathbb{R}^C$, which corresponds to the C target classes.

To be more specific, Eqs. (15)–(17) present the underlying operations with respect to the models LSTM, BiLSTM and GRU, respectively.

$$\begin{aligned} h^{(i)} &= LSTM(V^{(i)}) \\ dl^{(i)} &= dense(h^{(i)}) \\ \hat{p}^{(i)} &= softmax(dl^{(i)}),\ \hat{p}^{(i)} \in \mathbb{R}^C \\ M_{LSTM}(V^{(i)}) &= \arg\max_{c \in C}(\hat{p}^{(i)}) \end{aligned} \tag{15}$$

$$\begin{aligned} h^{(i)} &= BiLSTM(V^{(i)}) \\ dl^{(i)} &= dense(h^{(i)}) \\ \hat{p}^{(i)} &= softmax(dl^{(i)}),\ \hat{p}^{(i)} \in \mathbb{R}^C \\ M_{BiLSTM}(V^{(i)}) &= \arg\max_{c \in C}(\hat{p}^{(i)}) \end{aligned} \tag{16}$$

$$\begin{aligned} h^{(i)} &= GRU(V^{(i)}) \\ dl^{(i)} &= dense(h^{(i)}) \\ \hat{p}^{(i)} &= softmax(dl^{(i)}),\ \hat{p}^{(i)} \in \mathbb{R}^C \\ M_{GRU}(V^{(i)}) &= \arg\max_{c \in C}(\hat{p}^{(i)}) \end{aligned} \tag{17}$$

According to these models, we distinguish three different architectures denoted A_{LSTM}, A_{BiLSTM} and A_{GRU}. Each one consists of a set of k models represented as follows:

$$A_{LSTM} = (M^{(1)}_{LSTM}, \cdots, M^{(k)}_{LSTM}) \tag{18}$$

$$A_{BiLSTM} = (M^{(1)}_{BiLSTM}, \cdots, M^{(k)}_{BiLSTM}) \tag{19}$$

$$A_{GRU} = (M^{(1)}_{GRU}, \cdots, M^{(k)}_{GRU}) \tag{20}$$

where $M^{(j)}_{LSTM}$, $M^{(j)}_{BiLSTM}$, $M^{(j)}_{GRU}$ denotes the j^{th} LSTM, BiLSTM, GRU model, respectively.

The underlying rationale behind adopting one of the Eqs. (18)–(20) for representing sequential data is the assumption that, enhancing data representation plays a fundamental role for boosting the performance of classifiers, as well as overcoming the weaknesses of existing research works.

In particular, after training the set of k distributed models of an architecture A_{LSTM}, the values estimated by each model $M^{(j)}_{LSTM}$ are combined to produce an optimal representation. Thus, the input sequence $V^{(i)}$ can be re-represented as follows:

$$O^{(i)} = (M^{(1)}_{LSTM}(V^{(i)}), \cdots, M^{(k)}_{LSTM}(V^{(i)})) \tag{21}$$

where $M^{(1)}_{LSTM}(V^{(i)}) \in \mathbb{R}$, and $O^{(i)} \in \mathbb{R}^k$ stands for the new representation for the review $R^{(i)}$.

Similarly, in case of using an ensemble A_{BiLSTM} or A_{GRU}, the optimal representations are formulated as follows:

$$O^{(i)} = (M^{(1)}_{BiLSTM}(V^{(i)}), \cdots, M^{(k)}_{BiLSTM}(V^{(i)})) \tag{22}$$

$$O^{(i)} = (M^{(1)}_{GRU}(V^{(i)}), \cdots, M^{(k)}_{GRU}(V^{(i)})) \tag{23}$$

4.2.4. Text classification

In general, text classification is defined as the problem of learning classification model from reviews with pre-defined labels. Then, the learned model is employed to classify reviews in the future.

Logistic Regression is one of the well-known machine learning algorithms. It is a supervised learning method, which is commonly used for classification tasks. In this step, after the representation process, the training set can be represented as follows:

$$T = \{(x_i,\ y_i),\ i = 1, \cdots, |T|\},\ x_i \in \mathbb{R}^k \tag{24}$$

where $x_i = O^{(i)}$ stands for the estimated representation (i.e., the features), and y_i is the class label associated to the review $R^{(i)}$.

The main objective is to train a multinomial logistic regression model that can take a vector $O^{(i)} \in \mathbb{R}^k$ as input, and predict the probabilities of the different possible classes.

Using the trained model, the probability that x_i belongs to the class label y_i is expressed as follows (Cawley, Talbot, & Girolami, 2007; Krishnapuram, Carin, Figueiredo, & Hartemink, 2005):

$$P(y_i^{(c)} = 1 \mid x_i; \theta) = \frac{\exp(\theta^{(c)^T} x_i)}{\sum_{j=1}^{C} \exp(\theta^{(j)^T} x_i)} \tag{25}$$

where $c \in \{1, \cdots, C\}$, $y_i^{(c)}$ equals 1 if the class label is c, otherwise, it is equal to 0. $\theta^{(c)}$ denotes the parameter vector corresponding to the class c. T denotes the matrix transpose, and θ represents all the parameters of the model.

Finally, based on the estimated probabilities, the model generates the predicted class.

5. Experiments

5.1. Data sets

In this work, the experiments are conducted using two real-world data sets, namely, Yelp and Sentiment140.

- **Yelp**: This data set is composed 6,685,900 classified reviews provided by 1,637,138 users for 192,609 businesses. In this work, we have randomly selected 100,000 reviews as the original data set. We have considered 1 and 2 stars as a negative class, while 4 and 5 as a positive class (Yelp, 2019).
- **Sentiment140**: This is a Twitter sentiment analysis data set, which is originated from Stanford University. This particular data set consists of 1,600,000 classified tweets. In this work, we have randomly selected 20,000 tweets as the original data set (Sentiment140, 2019).

5.2. Evaluation metrics

To compare the performance of classifiers, we have adopted two metrics, namely, accuracy and F-score. They are widely used to assess the performance of sentiment analysis systems (Abdi, Shamsuddin, Hasan, & Piran, 2019; Al-Smadi, Al-Ayyoub, Jararweh, & Qawasmeh, 2019). The Accuracy and F-score metrics are defined as follows:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{26}$$

$$\begin{aligned} Precision &= \frac{TP}{TP + FP} \\ Recall &= \frac{TP}{TP + FN} \\ F - score &= \frac{2{*}Precision{*}Recall}{Precision + Recall} \end{aligned} \tag{27}$$

where

- **TN** is the number of true positives.
- **TP** refers to the number of true positives.
- **FP** denotes the number of false positives
- **FN** is the number of false negatives.

5.3. Methodology

For the experimental study, each data set is partitioned into two parts. The training set is composed of 90% of data, while the test set contains 10%. The training set is used for training the model, and the test set for assessing the performance. This procedure is repeated 5 times. Then the average is computed.

5.4. Implementation detail

The experiments have been executed on a cluster, which is composed of two nodes: one master node, and a slave node.

In addition, we have used fastText[1] for distributed word representations. The deep learning models were written in Python. Details of the various frameworks adopted in this work are as follows:

- PySpark 2.3.3
- Cassandra 3.0.15
- Kafka 2.1.0
- BigDL 0.7.0

[1] https://fasttext.cc/

- Spark MLLib
- Kibana 6.6.2
- Elasticsearch 6.6.2
- Python 3.5
- Ubuntu 16.04 LTS

5.5. Parameter setting

An extensive number of experimentations were conducted to determine the best settings. The parameters used for each method are presented as follows.

In general, the different deep learning models were optimized using Adam optimization algorithm (Kingma & Ba, 2015). The batch size was set to 64. The dropout (Hinton et al., 2012) rate for the dense layer was fixed at 0.5, while the recurrent dropout was set to 0.2. The parameter of representation $k = 9$. The learning rate was fixed at 0.01.

5.6. Experimental results

This section presents the experiments carried out using two real-world data sets. Particularly, our proposal is compared with the following deep learning approaches:

- LSTM: Long short-term memory model with the fastText word representations.
- BiLSTM (Graves et al., 2013): Bidirectional long short-term memory model with the pre-trained fastText word representations.
- GRU (Cho et al., 2014; Chung et al., 2014): Gated recurrent unit model with the pre-trained fastText word representations.
- XGBoost-avg-fastText (Chen & Guestrin, 2016): Extreme gradient boosting algorithm based on the average of fastText word embeddings.
- RF-avg-fastText: Random forest classifier based on the average of fastText word embeddings.
- Our proposal-LSTM: The proposed distributed method based on LSTM layer and fastText word representations.
- Our proposal-BiLSTM: The proposed distributed method based on BiLSTM layer and fastText word representations.
- Our proposal-GRU: The proposed distributed method based on GRU layer and fastText word representations.

Tables 1 and 2 present the experimental results on the sentiment140 and Yelp data sets, respectively. Each table reports the comparative results in terms of accuracy and F-score of our model against well-known deep learning approaches used for sentiment analysis task. To highlight the best result, we mark in bold the best score.

Table 1
Experimental results on sentiment140 data set (average for 5 runs).

Model	Accuracy	*F*-score
RF-avg-fastText	0.7265	0.7227
XGBoost-avg-fastText	0.736	0.7297
LSTM	0.7745	0.7777
BiLSTM	0.7792	0.7771
GRU	0.7826	0.7882
Our proposal-LSTM	**0.7809**	**0.7843**
Our proposal-BiLSTM	**0.7876**	**0.7865**
Our proposal-GRU	**0.7888**	**0.7902**

Table 2
Experimental results on Yelp data set (average for 5 runs).

Model	Accuracy	*F*-score
RF-avg-fastText	0.8453	0.8464
XGBoost-avg-fastText	0.856	0.8562
LSTM	0.9174	0.9191
BiLSTM	0.9229	0.9237
GRU	0.9239	0.9219
Our proposal-LSTM	**0.9255**	**0.9269**
Our proposal-BiLSTM	**0.9291**	**0.9295**
Our proposal-GRU	**0.9328**	**0.9310**

From Tables 1 and 2, it is obvious that our proposal shows higher accuracy and *F*-score than other state-of-the-art methods. It substantially outperforms the other competitors on both data sets. In particular, compared with the recurrent neural network variants, Distributed Long short-term memory (LSTM), Distributed Bidirectional long short-term memory (BiLSTM) and Distributed Gated recurrent unit (GRU) models, the obtained results demonstrate that the improved methods (i.e., Our proposal-LSTM, Our proposal-BiLSTM, Our proposal-GRU) yield higher accuracy and *F*-score in comparison with the original methods.

In view of the obtained results, it is reasonable to conclude that our proposal is able to enhance the performance of several existing works and thus contribute to more efficient models for big data sentiment analysis.

6. Conclusions

With the advent of social media applications, social big data has become an important topic for a large number of research areas. In this paper, we have proposed a distributed intelligent system for real-time social big data analytics. It is primarily designed based on a set of powerful big data tools to manage and process, and analyze large-scale data efficiently, as well as, improve real-time decision-making processes. In addition, we have devised an effective solution to improve the performance of a set of distributed deep learning models with fastText for sentiment analysis.

Extensive experiments demonstrate the promise of our work. In particular, the results show that our proposed solution is able to increase the accuracy of well-known deep learning models (i.e., Long short-term memory (LSTM), Bidirectional long short-term memory (BiLSTM) and Gated recurrent unit (GRU)).

Furthermore, the current work can provide many benefits for practitioners and researchers who want to collect, handle, analyze and visualize several sources of information in real-time. Also, it contributes to a better understanding of public opinion and user behavior, thanks to the improved variants of the most powerful distributed deep learning and machine learning algorithms. Moreover, it allows improving several existing works based on RNN variants for natural language processing.

For future work, we plan to investigate the impact of incorporating additional information into fastText vector representations for solving the sentiment analysis problem in the context of big data. As a second future work, it would be interesting to develop other strategies to further improve the performance of distributed deep learning models for natural language processing.

Acknowledgments

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

References

Abdi, A., Shamsuddin, S. M., Hasan, S., & Piran, J. (2019). Deep learning-based sentiment classification of evaluative text based on multi-feature fusion. *Information Processing & Management, 56*(4), 1245–1259.

Ait Hammou, B., Ait Lahcen, A., & Mouline, S. (2018). Apra: An approximate parallel recommendation algorithm for big data. *Knowledge-Based Systems, 157*, 10–19.

Al-Smadi, M., Al-Ayyoub, M., Jararweh, Y., & Qawasmeh, O. (2019). Enhancing aspect-based sentiment analysis of arabic hotels' reviews using morphological, syntactic and semantic features. *Information Processing & Management, 56*(2), 308–319.

Alharbi, A. S. M., & de Doncker, E. (2019). Twitter sentiment analysis with a deep neural network: An enhanced approach using user behavioral information. *Cognitive Systems Research, 54*, 50–61.

Bello-Orgaz, G., Jung, J. J., & Camacho, D. (2016). Social big data: Recent achievements and new challenges. *Information Fusion, 28*, 45–59.

BigDL (2019). Distributed deep learning library for apache spark. (Accessed: 10 April 2019) URL: https://software.intel.com/en-us/ai/frameworks/bigdl.

Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics, 5*, 135–146.

Cassandra (2019). Apache cassandra. (Accessed: 10 April 2019) URL: http://cassandra.apache.org/.

Cawley, G. C., Talbot, N. L., & Girolami, M. (2007). *Sparse multinomial logistic regression via Bayesian l1 regularisation. Advances in neural information processing systems*209–216.

Chen, M., Mao, S., & Liu, Y. (2014). Big data: A survey. *Mobile Networks and Applications, 19*(2), 171–209. https://doi.org/10.1007/s11036-013-0489-0.

Chen, T., & Guestrin, C. (2016). *Xgboost: A scalable tree boosting system. Proceedings of the 22nd ACM SIGKDDinternational conference on knowledge discovery and data mining.* ACM785–794.

Chen, T., Xu, R., He, Y., & Wang, X. (2017). Improving sentiment analysis via sentence type classification using BILSTM-CRF and CNN. *Expert Systems with Applications, 72*, 221–230.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., & Schwenk, H. (2014). *Learning phrase representations using RNN encoder–decoder for statistical machine translation. Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics1724–1734. https://doi.org/10.3115/v1/D14-1179.

Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). *Empirical evaluation of gated recurrent neural networks on sequence modeling. Nips 2014 workshop on deep learning, December 2014*.

Del Vecchio, P., Mele, G., Ndou, V., & Secundo, G. (2018). Creating value from social big data: implications for smart tourism destinations. *Information Processing & Management, 54*(5), 847–860.

Elasticsearch (2019). Elasticsearch. (Accessed: 10 April 2019) URL: https://www.elastic.co/products/elasticsearch.

FastText (2019). Fasttext: Library for efficient text classification and representation learning. (Accessed: 10 April 2019) URL: https://fasttext.cc/.

Gandomi, A., & Haider, M. (2015). Beyond the hype: Big data concepts, methods, and analytics. *International Journal of Information Management, 35*(2), 137–144.

García-Gil, D., Luengo, J., García, S., & Herrera, F. (2019). Enabling smart data: Noise filtering in big data classification. *Information Sciences, 479*, 135–152.

García-Gil, D., Ramírez-Gallego, S., García, S., & Herrera, F. (2017). A comparison on scalability for batch big data processing on apache spark and apache flink. *Big Data Analytics, 2*(1), 1.

Gormley, C., & Tong, Z. (2015). *Elasticsearch: The definitive guide: A distributed real-time search and analytics engine.* O'Reilly Media, Inc.

Graves, A., Mohamed, A., & Hinton, G. (2013). *Speech recognition with deep recurrent neural networks. 2013 IEEE international conference on acoustics, speech and signal processing*6645–6649. https://doi.org/10.1109/ICASSP.2013.6638947.

Graves, A., & Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks, 18*(5–6), 602–610.

Greff, K., Srivastava, R. K., Koutník, J., Steunebrink, B. R., & Schmidhuber, J. (2017). Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems, 28*(10), 2222–2232.

Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.

Hu, H., Wen, Y., Chua, T.-S., & Li, X. (2014). Toward scalable systems for big data analytics: A technology tutorial. *IEEE Access, 2*, 652–687.

Iyyer, M., Manjunatha, V., Boyd-Graber, J., & Daumé III, H. (2015). *Deep unordered composition rivals syntactic methods for text classification. Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)1. Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)* 1681–1691.

Jan, B., Farman, H., Khan, M., Imran, M., Islam, I. U., Ahmad, A., et al. (2019). Deep learning in big data analytics: A comparative study. *Computers & Electrical Engineering, 75*, 275–287.

Jianqiang, Z., & Xiaolin, G. (2017). Comparison research on text pre-processing methods on twitter sentiment analysis. *IEEE Access, 5*, 2870–2879.

Jianqiang, Z., Xiaolin, G., & Xuejun, Z. (2018). Deep convolution neural networks for twitter sentiment analysis. *IEEE Access, 6*, 23253–23260.

Jimenez-Marquez, J. L., Gonzalez-Carrasco, I., Lopez-Cuadrado, J. L., & Ruiz-Mezcua, B. (2019). Towards a big data framework for analyzing social media content. *International Journal of Information Management, 44*, 1–12.

Kafka (2019). Apache kafka. (Accessed: 10 April 2019). URL:https://kafka.apache.org/.

Kalchbrenner, N., Grefenstette, E., & Blunsom, P. (2014). *A convolutional neural network for modelling sentences. Proceedings of the 52nd annual meeting of the association for computational linguistics (volume 1: Long papers)*. Baltimore, Maryland: Association for Computational Linguistics655–665. https://doi.org/10.3115/v1/P14-1062.

Kibana (2019). Kibana. (Accessed: 10 April 2019). URL:https://www.elastic.co/products/kibana.

Kim, Y. (2014). *Convolutional neural networks for sentence classification. Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics1746–1751. https://doi.org/10.3115/v1/D14-1181.

Kim, Y., Jernite, Y., Sontag, D., & Rush, A. M. (2016). *Character-aware neural language models. Thirtieth AAAI conference on artificial intelligence*.

Kingma, D. P., & Ba, J. (2015). *Adam: A method for stochastic optimization. 3rd international conference on learning representations (ICLR)*.

Kreps, J., Narkhede, N., Rao, J., et al. (2011). *Kafka: A distributed messaging system for log processing. Proceedings of the NETDB*1–7.

Krishnapuram, B., Carin, L., Figueiredo, M. A., & Hartemink, A. J. (2005). Sparse multinomial logistic regression: Fast algorithms and generalization bounds. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*(6), 957–968.

Lakshman, A., & Malik, P. (2010). Cassandra: A decentralized structured storage system. *ACM SIGOPS Operating Systems Review, 44*(2), 35–40.

Lee, I. (2017). Big data: Dimensions, evolution, impacts, and challenges. *Business Horizons, 60*(3), 293–303.

Lin, B. Y., Xu, F., Luo, Z., & Zhu, K. (2017). *Multi-channel BILSTM-CRF model for emerging named entity recognition in social media. Proceedings of the 3rd workshop on noisy user-generated text*160–165.

Liu, G., & Guo, J. (2019). Bidirectional LSTM with attention mechanism and convolutional layer for text classification. *Neurocomputing, 337*, 325–338.

Maas, A. L., Hannun, A. Y., & Ng, A. Y. (2013). *Rectifier nonlinearities improve neural network acoustic models. Proceedings of the 30th international conference on machine learning (ICML-13)*.

McAfee, A., Brynjolfsson, E., Davenport, T. H., Patil, D., & Barton, D. (2012). Big data: the management revolution. *Harvard Business Review, 90*(10), 60–68.

Meng, X., Bradley, J., Yavuz, B., Sparks, E., Venkataraman, S., Liu, D., et al. (2016). Mllib: Machine learning in apache spark. *The Journal of Machine Learning Research, 17*(1), 1235–1241.

Mikolov, T., Grave, E., Bojanowski, P., Puhrsch, C., & Joulin, A. (2018). *Advances in pre-training distributed word representations. Proceedings of the international conference on language resources and evaluation (LREC 2018)*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). *Distributed representations of words and phrases and their compositionality. Proceedings of the 26th international conference on neural information processing systems – volume 2NIPS'13*3111–3119.

MLlib (2019). Machine learning library (mllib). (Accessed: 10 April 2019). URL: https://spark.apache.org.

Mohammadi, M., Al-Fuqaha, A., Sorour, S., & Guizani, M. (2018). Deep learning for IoT big data and streaming analytics: A survey. *IEEE Communications Surveys & Tutorials, 20*(4), 2923–2960.

Najafabadi, M. M., Villanustre, F., Khoshgoftaar, T. M., Seliya, N., Wald, R., & Muharemagic, E. (2015). Deep learning applications and challenges in big data analytics. *Journal of Big Data, 2*(1), 1.

Ngai, E. W., Tao, S. S., & Moon, K. K. (2015). Social media research: Theories, constructs, and conceptual frameworks. *International Journal of Information Management, 35*(1), 33–44.

Pennington, J., Socher, R., & Manning, C. (2014). *Glove: Global vectors for word representation. Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*1532–1543.

Pham, D.-H., & Le, A.-C. (2018). Exploiting multiple word embeddings and one-hot character vectors for aspect-based sentiment analysis. *International Journal of Approximate Reasoning, 103*, 1–10.

Ragini, J. R., Anand, P. R., & Bhaskar, V. (2018). Big data analytics for disaster response and recovery through sentiment analysis. *International Journal of Information Management, 42*, 13–24.

Rezaeinia, S. M., Rahmani, R., Ghodsi, A., & Veisi, H. (2019). Sentiment analysis based on improved pre-trained word embeddings. *Expert Systems with Applications, 117*, 139–147.

Saggi, M. K., & Jain, S. (2018). A survey towards an integration of big data analytics to big insights for value-creation. *Information Processing & Management, 54*(5), 758–790.

Salehan, M., & Kim, D. J. (2016). Predicting the performance of online consumer reviews: A sentiment mining approach to big data analytics. *Decision Support Systems, 81*, 30–40.

Sentiment140 (2019) data set. (Accessed: 2 March 2019) URL: http://cs.stanford.edu/people/alecmgo/trainingandtestdata.zip.

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., & Manning, C. D. (2011). *Semi-supervised recursive autoencoders for predicting sentiment distributions. Proceedings of the conference on empirical methods in natural language processing*. Association for Computational Linguistics151–161.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., & Potts, C. (2013). *Recursive deep models for semantic compositionality over a sentiment treebank. Proceedings of the 2013 conference on empirical methods in natural language processing*1631–1642.

Song, M., Park, H., & Shin, K.-S. (2019). Attention-based long short-term memory network using sentiment lexicon embedding for aspect-level sentiment analysis in korean. *Information Processing & Management, 56*(3), 637–653.

Spark (2019). Apache spark. (Accessed: 10 April 2019) URL:https://spark.apache.org/.

Stieglitz, S., Mirbabaie, M., Ross, B., & Neuberger, C. (2018). Social media analytics–challenges in topic discovery, data collection, and data preparation. *International Journal of Information Management, 39*, 156–168.

Sutskever, I., Martens, J., & Hinton, G. E. (2011). *Generating text with recurrent neural networks. Proceedings of the 28th international conference on machine learning (ICML-11)*1017–1024.

Tai, K. S., Socher, R., & Manning, C. D. (2015). Improved semantic representations from tree-structured long short-term memory networks,. (pp. 1556–1566) URL: https://www.aclweb.org/anthology/P15-1150.10.3115/v1/P15-1150.

Tien, N. H., & Le, N. M. (2017). *An ensemble method with sentiment features and clustering support. Proceedings of the eighth international joint conference on natural language processing (volume 1: Long papers)1. Proceedings of the eighth international joint conference on natural language processing (volume 1: Long papers)* 644–653.

Valdivia, A., Luzón, M. V., & Herrera, F. (2017). Sentiment analysis in tripadvisor. *IEEE Intelligent Systems, 32*(4), 72–77.

Vosoughi, S., Vijayaraghavan, P., & Roy, D. (2016). *Tweet2vec: Learning tweet embeddings using character-level CNN-LSTM encoder-decoder. Proceedings of the 39th*

international ACMSIGIR conference on research and development in information retrieval. ACM1041–1044.
Wang, J., Yu, L.-C., Lai, K. R., & Zhang, X. (2016). *Dimensional sentiment analysis using a regional CNN-LSTM model. Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers)2. Proceedings of the 54th annual meeting of the association for computational linguistics (volume 2: Short papers)* 225–230.
Wang, X., Liu, Y., Chengjie, S., Wang, B., & Wang, X. (2015). *Predicting polarities of tweets by composing word embeddings with long short-term memory. Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)1. Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)* 1343–1353.
Wang, Y., Kung, L., & Byrd, T. A. (Kung, Byrd, 2018a). Big data analytics: Understanding its capabilities and potential benefits for healthcare organizations. *Technological Forecasting and Social Change, 126*, 3–13.
Wang, Y., Qiu, X., Ding, D., Zhang, Y., Wang, Y., Jia, X., et al. (2018). Bigdl: A distributed deep learning framework for big data. arXiv preprint arXiv:1804.05839.
Xiang, Z., Schwartz, Z., Gerdes Jr, J. H., & Uysal, M. (2015). What can big data and text analytics tell us about hotel guest experience and satisfaction? *International Journal of Hospitality Management, 44*, 120–130.
Yelp (2019). Yelp data set. (Accessed: 2 March 2019) URL: https://www.yelp.com/dataset/challenge.
Yu, L.-C., Wang, J., Lai, K. R., & Zhang, X. (2017). *Refining word embeddings for sentiment analysis. Proceedings of the 2017 conference on empirical methods in natural language processing*534–539.
Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., et al. (2012). *Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. Proceedings of the 9th USENIX conference on networked systems design and implementationNSDI'12*Berkeley, CA, USA: USENIX Association.
Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., & Stoica, I. (2010). *Spark: Cluster computing with working sets. Proceedings of the 2nd usenix conference on hot topics in cloud computingHotCloud'10*Berkeley, CA, USA: USENIX Association 10–10.
Zhang, Q., Yang, L. T., & Chen, Z. (2016). Deep computation model for unsupervised feature learning on big data. *IEEE Transactions on Services Computing, 9*(1), 161–171.
Zhang, Q., Yang, L. T., Chen, Z., & Li, P. (2018). A survey on deep learning for big data. *Information Fusion, 42*, 146–157.