Journal of Artificial Intelligence and Data Mining (JAIDM), x(x): xxx-xxx, xxxx

Shahrood University of Technology

Journal of Artificial Intelligence and Data Mining (JAIDM)
Journal homepage: http://jad.shahroodut.ac.ir

Research paper

Distributed online pre-processing framework for big data sentiment analytics

Mahdi Molaei and Davud Mohammadpur*

Department of Computer Engineering, University of Zanjan, Iran.

Article Info

Article History:
Received 28 October 2021
Revised 04 January 2022
Accepted 16 February 2022

DOI:10.22044/jadm.2022.11330.2293

Keywords:
Big data, Sentiment analysis, pre-processing, Apache Spark, DataFrame, Recurrent Neural Networks.

**Corresponding author: dmp@znu.ac.ir (D. Mohammadpour).*

Abstract

Performing sentiment analysis on the social network big data can be helpful for various research and business projects to take useful insights from the text-oriented content. In this work, we propose a general pre-processing framework for sentiment analysis, which is devoted to adopt FastText with recurrent neural network variants in order to prepare textual data efficiently. This framework consists of three different stages of data cleansing, tweets padding, word embedding's extraction from FastText, and conversion of tweets to these vectors, which are implemented using the DataFrame data structure in Apache Spark. Its main objective is to enhance the performance of online sentiment analysis in terms of the pre-processing time, and handle a large-scale data volume. In addition, we propose a distributed intelligent system for the online social big data analytics. It is designed to store, process, and classify a huge amount of information online. The proposed system adopts any word embedding libraries like FastText with different distributed deep learning models like LSTM. The results of the evaluations show that the proposed framework can significantly improve the performance of the previous RDD-based methods in terms of the processing time and data volume.

1. Introduction

Today, social networks have a significant role in the people's daily lives, and a lot of data is generated daily in these networks. This big data generated by social networks is a valuable data source for various research projects, especially in marketing, and allows the companies to analyze the customer opinions about their products or services, and make strategic decisions for the future [1].

Sentiment analysis is one of the fundamental tasks in natural language processing for analyzing and recognizing the polarity of text. In the recent years, the accuracy of these systems using deep learning models has significantly increased [2]. However, with increasing data volume and other factors such as using old data structures, efficiency of the traditional systems in storing and processing these data has been decreased. It implies the importance of developing a distributed framework using a new platform.

Besides, the online processing system design is a complex process that involves several challenges. The processing time and processing data volume are two essential factors in the efficiency of this type of system. Therefore, it is important to solve the challenges related to these two factors in designing an online sentiment analysis system [3].

Also the big data processing frameworks has created a new opportunity to process and extract more in-depth knowledge in various fields including business, health, and smart city, which integrates with the machine learning techniques. In most machine learning methods, by considering more amount of data, the accuracy of the results is increased since more variations of the problem can be observed and applied to the model [4, 5].

With the advancement of various technologies such as the web technologies, social networks and sensors, the volume and speed of data production in various formats are increasing unprecedentedly. Twitter, for example, processes over 500 million tweets per day, generating more than eight terabytes of data per day [6, 7]. However, the current implementations of machine learning methods could not store, manage, and analyze this data. Therefore, new platforms and processing systems are required for these goals [8]. For this reason, the Apache Hadoop platform was introduced at first. This platform uses a distributed architecture to store data with the Map-Reduce execution model. However, this platform has limitations such as a drop in performance on repetitive algorithms such as machine learning algorithms, which lead to introduce new platforms such as Apache Spark to address its shortcomings.

1.1. Sentiment analysis

Sentiment analysis involves the process of automatically detecting the polarity of a text and extracting the author's reviews on the subject, and finally, classifying the text [9]. In many research approaches, the textual data classification is done using deep learning models. This is due to the ability of deep learning models to classify a text with a high accuracy and the ability to model the sequence of textual data with word dependencies throughout the sentence. One of these deep learning models is RNN (Recurrent Neural Network). In order to use these models, the textual data and words must be converted into numerical vectors, for which various algorithms and methods have been proposed [10]. Today's pre-trained word embedding libraries such as FastText have a high accuracy and quality in vector representations for words. Accordingly, in most current systems and research approaches, these libraries are used to convert the textual data to numerical vectors [11].

One important point in converting tweets to numeric vectors is to use an optimal pre-processing platform to clean and convert tweets to numeric vectors [12]. Implementation of such an optimal framework is a challenging issue due to the large volume and high production speed. For example, performing these processes using Map-Reduce require repeated Map-Reduce operations, which cause heavy computational overhead on the system, and so the processing time increases with increasing data volume with a very steep slope, that is not optimal at all [13].

1.2. Apache Spark data model

Apache Spark performs in-memory processing and computations, which speeds up to about 100 times faster than Hadoop, especially in the machine learning algorithm, and allows more data to be processed instantly. In Apache Spark, different libraries have been introduced for different purposes, enabling the platform to be usable in various contexts. Libraries such as MLlib and ML are used to implement machine learning algorithms, the GraphX library for graphic processing, and Spark Streaming used to process the streaming data [14].

Spark data structure plays a significant role in the fast and efficient big data processing. In general, a data structure is a data organization, management, and storage format that can enable an efficient access and modification. More precisely, a data structure is a collection of data values, the relationships among them, and the functions or operations applied to the data [15].

Resilient Distributed Datasets (RDD) is a fundamental data structure of Apache Spark. It is an immutable distributed collection of objects. Each dataset in RDD is divided into logical partitions, which may be computed on different nodes of the cluster [4, 14]. DataFrame is a distributed dataset based on RDD that organizes the data in name. It is similar to a 2D table in the relational database, so it introduces the database's schema. DataFrame is accompanied by richer optimizations in Apache Spark. Like RDD, it is a distributed dataset that has specific structures and named columns. When using the DataFrame, the optimizations in Spark are done by the Catalyst module. However, one of the disadvantages of the DataFrame data structure is the lack of storing the data type of each column [15, 16]. DataFrame has been designed from the ground-up to support modern big data and data science applications. As an extension to the existing RDD, DataFrames feature:

- Ability to scale from kilobytes of data on a single laptop to petabytes on a large cluster.
- Support for a wide array of data formats and storage systems.
- State-of-the-art optimization through the Spark SQL Catalyst optimizer.
- Seamless integration with all big data tooling and infrastructure via Spark.
- APIs for Python, Java, Scala, and R.

2. Related Works

Various applications for big data analysis have been proposed using machine learning methods, each of which has taken different approaches and

used different platforms and algorithms for this purpose. Previous research in various pre-processing and sentiment analysis techniques has been categorized on Twitter data, which we discuss in this section.

2.1. Text pre-processing techniques

The textual data generated on various social networks such as Twitter has many structural and spelling mistakes that are necessary to perform a series of pre-processing techniques to clear and correct them before classifying them. For this reason, various research works have been presented to introduce different techniques for doing this work, and examine the effectiveness of these techniques, which we will discuss in the following.

The impact of pre-processing on the sentiment analysis of Twitter data is considered in [11]. The main focus is on the tweets that use more symbols, abbreviations, and anonymous words. This method relies on the bindings of slang words on other coexisting words in order to check the slang word's significance and sentiment translation. They have used n-gram to find the bindings and conditional random fields in order to check the slang words' significance.

In [17], the effect of different pre-processing methods on Twitter sentiment analysis has been studied, and the results on the Stanford Twitter dataset have been evaluated. This study shows that deleting URLs, correcting words and negative verbs, and normalizing word characters positively affect sentiment analysis but stemming and lemmatization negatively affect and reduce the model's accuracy. In this model, a linear model is used to analyze sentiments.

In [8], the role of pre-processing on the film review dataset has been studied. In this research work, the pre-processing techniques such as expanding abbreviated terms, removing non-alphabetic symbols, correcting negative verbs by adding "NOT_" to their beginning, and stemming have been used. The TF_IDF method has also been used to convert text to vector, and the SVM algorithm has been used to perform the classification. This study shows that the use of appropriate pre-processing techniques including structure modification and word filtering can significantly increase the accuracy and quality of the model.

In [13], the pre-processing techniques on email and news datasets have been reviewed. In this research work, the techniques such as removing redundant words, converting words to lower case, and stemming have been used. This study has pointed out that there is no set of specific and unique techniques effective in all datasets and languages. The selection of these techniques should be made by identifying the characteristics of language and data carefully. The evaluation results indicate that the accuracy of the model increases by expanding the abbreviations and converting the negative verbs. Other techniques such as removing URLs, redundant numbers, and stop words will reduce model accuracy due to useful information in some of these cases.

2.2. Twitter sentiment analysis

Today, as the social networks' role in people's lives has become more prominent, the data generated on these networks daily has greatly increased. This data has become valuable resources for different aims such as predicting trends, sentiment analysis, marketing, and advertising. The following is a review of some of the research approaches on the Twitter sentiment analysis.

[6] provides a framework based on Apache Hadoop and Mahout in order to analyze tweets' sentiments about the "Airtel" service. In this framework, the tweets published with the hashtag #airtel_presence are first collected and stored in Hadoop HDFS. Next, after pre-processing, the tweets are converted into word embedding using the TF-IDF method. The tweets are then classified into three classes: positive, negative, and neutral, using the Naïve Bayes in Apache Mahout.

In [3], an approach to sentiment analysis based on Twitter data uses the integration of machine learning algorithms and streaming analysis approaches. It allows the researchers in this field to extract valuable information from tweets instantly and classify them without worrying about the time and memory required. The advantages of the method proposed in this paper include the possibility of building a larger dictionary of the desired features, processing larger volumes of data, and online prediction based on machine learning algorithms.

In [5], the Apache Spark-based framework for online sentiment analysis of Twitter streaming data has been presented with the Fake Twitter Accounts Detection Service. This article uses the Spark MLlib library and Naïve Bayes algorithm to detect fake Twitter accounts and sentiment analysis. Finally, the evaluation of the above framework has been done by two methods. In the first method, data is used without pre-processing. However, in the second method, a series of pre-processing operations including marking, deleting stop words, stemming, and removing punctuation

are performed on the data, and then used to build a learning model.

Recently, new libraries have been introduced to implement deep learning models on Apache Spark. Although stable versions of these libraries have not yet been published, the use of these algorithms, in addition to the benefits of deep learning (such as improved accuracy) increases efficiency, and enables the processing of larger volumes of data. One of these libraries is BigDL, developed by Intel. In [1], an online framework for sentiment analysis on Twitter has been proposed using this library to implement deep learning models. In this article, the FastText library is used for word embedding. In this framework, tweets are collected with Apache Kafka and stored in Apache Cassandra. Then after pre-processing and extracting word embedding using FastText, the data is sent to the deep learning model for improving each word vector representation. Then tweets are classified with a logistic regression algorithm in Spark MLlib, and the results are indexed in Elasticsearch. Finally, the results are visually displayed by the Kibana Library. In this article, two datasets are used: "Yelp" with about one hundred thousand tweets and "Sentiment-140" with about twenty thousand tweets.

3. Proposed Method

In this work, we proposed a big data framework based on Apache Spark in order to determine whether the data is positive, negative or neutral. As mentioned earlier, the use of DataFrame is better than RDD in memory overhead and runtime due to the clear structure and named columns with specific optimizations. Most current methods and libraries are based on RDD but our proposed method is to replace the RDD data structure with DataFrame at various algorithm implementation stages. In this work, the main focus is on modifying the pre-processing algorithms but the same approach can be generalized to the libraries' main algorithms. The novelty of our approach consists of the migrating and extension of existing approaches from RDDs to DataFrames, enabling fast processing.

The pre-processing required for sentiment analysis consists of three steps. Each one of the steps is based on DataFrames, and is implemented in the form of transformers. A transformer is an algorithm that can convert one DataFrame to another one [12]. Transformer is one of the new Spark ML library components. The structure and operation of the transformer are shown in Figure 1. Since it is impossible to modify or edit the distributed data in Spark, a new data structure (DataFrame or RDD) must be created to convert or compute a new value; a concept called transformer is used to calculate and create a new version.

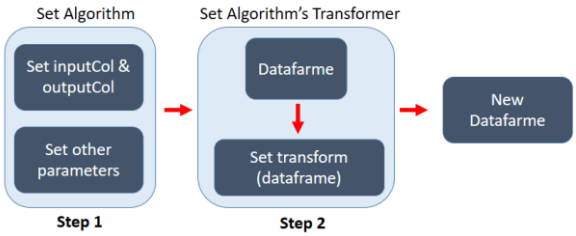

Figure 1. Architecture of transformer in Apache Spark.

3.1. Data cleansing

Applying a series of pre-processing on the textual data before analyzing is necessary and important in order to achieve an acceptable result. The pre-process aims to clarify and clear up the ambiguity of the input data for future analysis and classification. The techniques used in this study are as follows [9]:

Normalization: Tweets are required to be normalized first so that they can be used later. Sometimes, when some texts are very similar, they may be considered different by sentiment analysis due to some superficial differences. That is why the proposed method tries to eliminate such differences. The techniques used in this section are as follow:

• Converting words to lower case.
• Removing extra spaces.
• Deleting URLs, mentions, emojis, and numbers.

Stop words: Despite the repetition of these types of words, they are not conceptually effective for classifying the tweets' polarity. For example, the words such as "the" and "if" are of this type. Although it is assumed that only conjunctions are known as stop words, many auxiliary verbs, nouns, and adjectives can also be included in these terms, and have virtually no useful information for detecting the polarity of tweets. In most sentiment analysis systems, deleting these words increases the accuracy and performance of the system. Furthermore, deleting these words reduces the computational overhead and increases the speed. In our implementation, these words have been removed using the NLTK library.

Structural correction of words and tweets: In many words such as "coool," one or more letters are accidentally or intentionally repeated, which requires that these words are returned to their original form.

On the other hand, some words have spelling mistakes that are required to be corrected; otherwise, the model recognize them as a wrong word, and practically cannot create a vector. Also many of the verbs in these tweets are shortened, which must be placed in their complete form in order for the model to understand the concept and create a vector for them. For example, the verbs like "aren't" should be replaced with "are not."

Token construction: The sentiment analysis are required to be able to analyze and classify their data lexically. In the process of lexical analysis, each sentence is divided into a sequence of tokens. The tokens are the smallest meaningful part of a text. In general, this process is called tokenization, and ultimately leads to creating an independent set of meaningful parts of each text.

Previously, these processes were performed by Map-Reduce tasks on RDD data structure. In this research work, these processes were performed by replacing DataFrames instead of RDDs, and implemented by changing the algorithm in the form of a Transformer using the BigDL library, which causes a significant difference in the result.

3.2. Padding tweets

After the tweets are cleared, all tweets must be of the same length to feed them to the neural network model. Therefore, the number of words for each tweet and maximum length of tweets is calculated in this step. Then to the extent that it differs from the length of the largest tweet, meaningless words like "##" are added to them with a numeric vector of zero. Depending on the neural network architecture, this padding can be done from the right or left side of the tweets.

3.3. Extract word embedding vectors

In this section, we use the FastText pre-trained word vectors library provided by Facebook to convert tweets to vector representation. This library is based on the Skip-gram and Subword model, and contains 300-dimensional vectors of one million words. At this point, the tweets after tokenization and padding are sent to this transformer. In this transformer, in order to support all the pre-trained vector libraries, the word vectors are first extracted from the FastText model, and stored in a Python dictionary; then after extracting each word's numerical vectors, a 2D matrix is created for each tweet and stored in a column in the new DataFrame.

3.4. Sentiment analysis

After the pre-processing and converting the tweets to vectors, the sentiments are classified using the RNN model. RNN is a neural network model designed to model the sequential data. This model is generally used to perform various natural language processing tasks such as sentiment analysis, text classification, speech recognition, and machine translation [1]. In this part, we used the distributed GRU model on Apache Spark, which was implemented in the BigDL library.

4. Experiments

In this work, we used a real-world dataset named "Sentiment-140". This is a Twitter sentiment analysis dataset and used as a valid dataset in a variety of research works related to the Twitter sentiment analysis. This dataset contains 1,600,000 tweets stored in a CSV file. However, after removing the incomplete and empty tweets, the total number of tweets will be reduced to about 1,200,000. In this dataset, for each tweet, ID, date, query (lyx), author username, polarity, and text of tweet are stored. The polarity of each tweet is encoded using the numbers "0" for the negative tweets and "4" for the positive tweets.

Evaluations were performed in an Apache Spark cluster with a master machine and several slave nodes. Other specifications are one master machine with 6 GB memory and 4 processing cores along with several slave nodes with a total of 24 GB memory and 12 processing cores.

4.1. Evaluation metrics

In order to evaluate the proposed framework, we used the processing time, number of slave nodes, and various databases. We also used the accuracy (Equation 1) and F-score (Equation 2) in order to evaluate the performance of automatically detecting the polarity of a text. In general, the evaluation process was repeated five times, and the average results were calculated.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (1)$$

$$F - Score = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (2)$$

Where:

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

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

4.2. Experimental results

This section presents the experiments carried out using the Sentiment-140 dataset. Particularly, our proposal method is compared with the following deep learning approaches:

- GRU [18, 19]: Gated Recurrent Unit model with the pre-trained FastText word representations.
- XGBoost-avg [20]: Extreme Gradient Boosting algorithm based on the average of FastText word embedding.
- LSTM [21, 22]: Long Short-Term Memory model with the FastText word representations.
- RF-avg [1]: Random Forest classifier based on the average of FastText word embedding.
- Our proposal-method: The proposed distributed method based on the GRU and FastText word representations.

Table 1 presents the experimental results on the Sentiment-140 dataset. This table reports the comparative results in terms of accuracy and F-score of our model against the well-known deep learning approaches used for sentiment analysis task [1]. In order to highlight the best result, we mark in bold the best score.

Table 1. Experimental results on Sentiment-140 d.

	Accuracy	F-score
RF-avg	0.7265	0.7227
XGBoost-avg	0.736	0.7297
LSTM	0.7745	0.7777
GRU	0.7826	0.7882
Our proposed method	**0.7896**	**0.7973**

4.3. Evaluating proposed framework

As shown in Figure 2, the time chart of the proposed framework using the RDD data structure is an ascending diagram, and the Spark cluster is not able to execute more than 80,000 data and slave nodes crashed.

As it can be seen in the Figure 2, the results of the evaluation using the RDD data structure on the sorted data are similar to the unsorted state. However, when the data is sorted (sorting based on the length of tweets), additional processing overhead is imposed on the Spark cluster. Therefore, in the sorted mode, it is impossible to execute more than 60,000 data and cluster crashed.

Figure 3 shows the results of the evaluation using the DataFrame data structure on the sorted and unsorted data. As shown in Figure 3, the proposed framework was executed in a much shorter time than RDD-based framework, and in this case. In this case, the time chart is ascending, and in the range of 80,000 tweets, the runtime is almost equal, after which the runtime increases slightly.

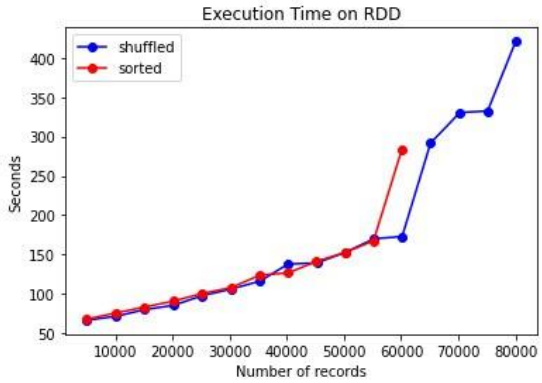

Figure 2. Evaluation of proposed framework based on RDD(million records).

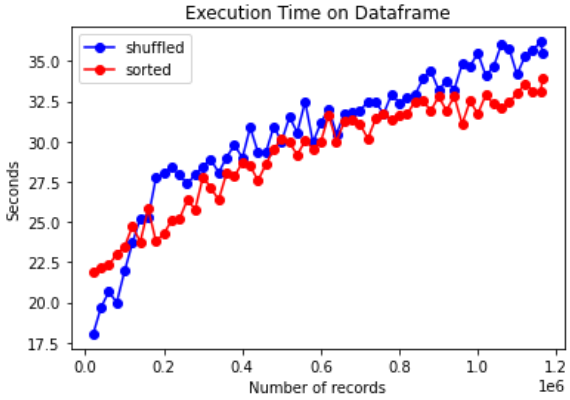

Figure 3. Evaluation of proposed framework based on DataFrame (million records).

In the sorted tweets, the execution time changes. In this case, up to 20,000 tweets, the execution of sorted data has a higher time but by increasing the data volume, the sorted data execution time is less than the unsorted. Based on this chart, it can be concluded that to process a big volume of tweets, sorting by length can reduce the execution time.

Figure 4 shows the evaluation results of the proposed framework based on RDD and DataFrame, integrated. As it can be seen, there is a huge difference in the execution time and volume between these two data structures.

4.4. Evaluating effect of different number of slave nodes on proposed framework

In this section, the effects of number of slave nodes were evaluated on the proposed framework. In the previous section, the number of slave nodes was set to 2 by default. In this section, the number of slave nodes was considered to be 3. The results obtained are as Figure 5.

The reason for this evaluation can be considered to the fact that by increasing the number of slave nodes, the master node can perform better and

more distribution on data and processes and execute processes in parallel, which reduces the required time to execute the processes.

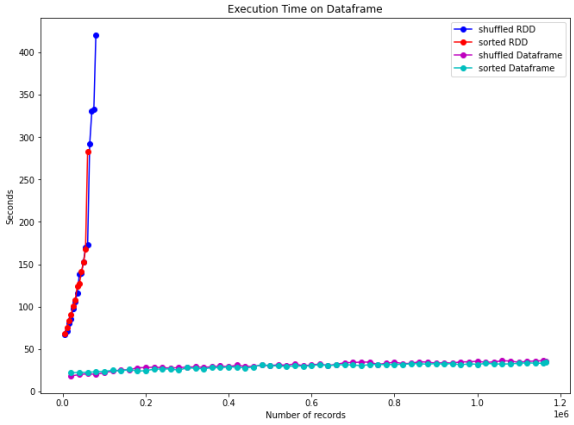

Figure 4. Evaluation of proposed framework based on RDD and DataFrame (million records).

Figure 5. Evaluation of proposed framework based on different number of slave nodes (million records).

4.5. Evaluating proposed framework based on different databases

Apache Spark is a lightning-fast cluster computer computing technology designed for fast computation, and also being widely used by industries. However, on the other side, it also has some ugly aspects. Apache Spark does not come with its own database management system. It depends on some other platforms. Thus Apache Spark has no database platform, and can only be used as a processing engine. However, the good thing about this is that Spark supports various databases as a storage layer. Spark can also retrieve and store data as a file but it cannot distribute that in the cluster. Therefore, if a file is used, the data file must be present in all machines' hard disk, which cannot store and retrieve massive data, and is not an optimal solution. For this reason, in this part of evaluation, two databases (Cassandra and MongoDB) are used as a storage layer of Spark, and their results are compared.

Both the Cassandra and MongoDB databases are NoSQL databases, except that Cassandra is column-oriented and MongoDB is a document-oriented database. The data can be distributed in both databases. Therefore, they can be the right choice as a storage layer in Spark. The results of this work can be seen in Figure 6.

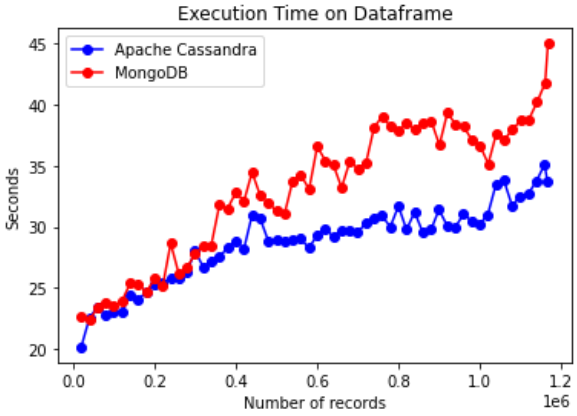

Figure 6. Evaluation of proposed framework based on different databases (million records).

In this comparison, the proposed framework is implemented based on DataFrame. The data is stored in each of the above two databases, and is retrieved and used during processing. As shown in Figure 6, the Apache Cassandra database performed better than the MongoDB to store and retrieve big data. Therefore, this database can be used as a suitable storage platform for big data.

4.6. Result analysis

As presented in the previous evaluations, we explored different options for creating a distributed platform and run different numbers of machines using different databases. Table 2 summarizes the evaluation results of the proposed framework. According to this table, the proposed framework based on DataFrame is suggested by the maximum number of machines and Apache Cassandra database to achieve the best result.

As shown in Table 2, average execution time, the integration of Apache Spark with proper databases such as Apache Cassandra can fully meet the big data processing requirements. Apache Cassandra also covers the weaknesses of file usage in Spark, and provides a better distribution. Using this database alongside Spark can significantly impact both the storage and retrieval of big data and the distribution of data.

Table 2. Summary of evaluation results of proposed framework.

Implementation method	Storage	Number of slave nodes	Data volume (Tweets)	Min-Max Execution time (s)	Average Execution time (s)
Proposed framework based on RDD-unsorted	CSV file	2 machines	5000 - 80000	66–420	264
Proposed framework based on RDD-sorted	CSV file	2 machines	5000 - 60000	67–425	281
Proposed framework based on DataFrame-unsorted	CSV file	2 machines	20000 - 1200000	18–36	29
Proposed framework based on DataFrame-sorted	CSV file	2 machines	20000 - 1200000	21–33	27
Proposed framework based on DataFrame	CSV file	1 machine	20000 - 1200000	22–37	32
Proposed framework based on DataFrame	CSV file	2 machines	20000 - 1200000	18–36	29
Proposed framework based on DataFrame	CSV file	3 machines	20000 - 1200000	18–33	27
Proposed framework based on DataFrame	Cassandra	2 machines	20000 - 1200000	20–33	**24**
Proposed framework based on DataFrame	MongoDB	2 machines	20000 - 1200000	22–45	26

Furthermore, the current work can provide many benefits for the practitioners and researchers who wish to collect, handle, and analyze big data sources of information online. In the results of this work, it was found that the proposed framework had an optimal performance compared to the previous methods, reducing the processing time and increasing the ability of large-scale data volume handling, based on which an online sentiment analysis system can be created.

5. Conclusion

As mentioned earlier, there may be various methods to perform big data pre-processing. However, in this work, we tried to implement and present the optimal method and framework based on Apache Spark. Basically, for processing large volumes of data, the big data processing tools can be handy, and improve efficiency.

This paper presents a DataFrame-based pre-processing framework to convert tweets to numeric vectors, and perform sentiment analysis on the Twitter data in order to increase the performance and support the big data features. The advantages of this framework include support for huge data volumes, reduced processing time required compared to the previous methods, and existence of a suitable storage layer to enable the storage and retrieval of distributed big data.

In particular, the results obtained show that our proposed solution is able to increase the accuracy of well-known deep learning models such as LSTM and GRU.

One of the ideas that can be considered for a future work is to implement a DataFrame-based processing platform for deep learning processes. Indeed, the processing of deep learning models is time-consuming, and usually takes a long time to complete the learning process. In Apache Spark, on the other hand, the processes are distributed. Therefore, implementing the deep learning processes in a distributed and database-based manner can reduce the model learning time and increase the system performance.

Another thing that can be considered in a future work is to implement an incremental model based on DataFrames for the stream data. With the entry of new data, it is unnecessary to perform all the processes from the beginning, and it is possible to process the new data separately and add their information to the model separately.

References

[1] B. Ait Hammou, A. Ait Lahcen, and S. Mouline, "Towards a real-time processing framework based on improved distributed Recurrent Neural Network variants with FastText for social big data analytics," *Information Processing and Management*, vol. 57, no. 1, pp. 102-122, 2020.

[2] H. Sadr, Mir M. Pedram, and M. Teshnehlab, "Convolutional Neural Network Equipped with Attention Mechanism and Transfer Learning for Enhancing Performance of Sentiment Analysis," *Journal of AI and Data Mining*, vol. 9, no. 2, pp. 141-151, 2021.

[3] A. Lakizadeh and Z. Zinaty, "A Novel Hierarchical Attention-based Method for Aspect-level Sentiment Classification," *Journal of AI and Data Mining*, vol. 9, no. 1, pp. 87-97, 2021.

[4] M.N. Farhan, H. Md Ahsan, and A. Md Arshad, "A study and performance comparison of mapreduce and apache spark on Twitter data on hadoop cluster," *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 10, no. 7, pp. 61-70, 2018.

[5] D. Kılınç, "A spark-based big data analysis framework for real-time sentiment prediction on streaming data," *Software: Practice and Experience*, vol. 49, no. 9, pp. 1352-1364, 2019.

[6] M. Kumar and B. Anju, "Analyzing Twitter sentiments through big data," in *2016 3rd International*

Conference on Computing for Sustainable Global Development (INDIACom), 2016, pp. 2628-2631, IEEE, 2016.

[7] A. L'heureux, K. Grolinger, and HF. Elyamany, "Machine learning with big data: Challenges and approaches." *IEEE Access*, vol. 5, no. 1, pp. 7776-7797, 2017.

[8] E. Haddi, X. Liu, and Y. Shi, "The role of text pre-processing in sentiment analysis," *Procedia Computer Science*, vol. 17, no. 1, pp. 26-32, 2013.

[9] M.K. Sohrabi, and F. Hemmatian, "An efficient pre-processing method for supervised sentiment analysis by converting sentences to numerical vectors: a twitter case study," *Multimedia tools and applications*, vol. 78, no. 17, pp. 24863-24882, 2019.

[10] M.W. Habib, and Z.N. Sultani, "Twitter Sentiment Analysis using Different Machine Learning and Feature Extraction Techniques," *Al-Nahrain Journal of Science*, vol. 24, no. 3, pp. 50-54, 2021.

[11] T. Singh and M. Kumari, "Role of text pre-processing in twitter sentiment analysis," *Procedia Computer Science*, vol. 89, no. 1, pp. 549-54, 2016.

[12] S. Symeonidis, D. Effrosynidis, and A. Arampatzis, "A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis," *Expert Systems with Applications*, vol. 110, no. 1, pp. 298-310, 2018.

[13] A.k. Uysal and S. Gunal, "The impact of pre-processing on text classification," *Information processing and management*, vol. 50, no. 1, pp. 104-12, 2014.

[14] M. Zaharia, R.S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M.J. Franklin, and A. Ghodsi, "Apache spark: a unified engine for big data processing," *Communications of the ACM*, vol. 59, no.11, pp. 56-65, 2016.

[15] J. Damji, "RDD *vs.* DataFrames and Datasets: A Tale of Three Apache Spark APIs," *Databriks Engineering Blog*, pp. 1-1, 2016. [Accessed Sept. 18, 2021].

[16] S. Salloum, R. Dautov, X. Chen, PX. Peng, and ZH. Joshua, "Big data analytics on Apache Spark," *International Journal of Data Science and Analytics*, vol. 1, no. 3, pp. 145-164, 2016.

[17] Y. Bao, C. Quan, L. Wang, and F. Ren, "The role of pre-processing in twitter sentiment analysis," in *International conference on intelligent computing*, Springer, pp. 615–624, 2014.

[18] J.Y. Cho and E.H. Lee, "Reducing confusion about grounded theory and qualitative content analysis: Similarities and differences," *Qualitative Report*, vol. 19, no. 32, pp. 1-15, 2014.

[19] A. Kumar, S. Abirami, T.E. Trueman, and E. Cambria, "Comment toxicity detection via a multichannel convolutional bidirectional gated recurrent unit," *Neurocomputing*, vol. 441, no. 1, pp.272-8, 2021.

[20] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," in *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pp. 785-794, 2016.

[21] BT. Hung BT, "Domain-specific versus general-purpose word representations in sentiment analysis for deep learning models," in *Frontiers in intelligent computing: Theory and applications*, Springer, pp. 252-264, 2020.

[22] F. Baratzadeh and Seyed M. H. Hasheminejad, "Customer Behavior Analysis to Improve Detection of Fraudulent Transactions using Deep Learning," *Journal of AI and Data Mining*, vol. 10, no. 1, pp. 1-16, 2022.

FastText

FastText

FastText

GRU LSTM

RDD