TAF **Journal of Advances in Technology and Engineering Research** **JATER**

2017, **3**(4): 108-116

ORIGINAL CONTRIBUTION

Analysis of real-time data with spark streaming

Nikitha Johnsirani Venkatesan [1]*, Choon Sung Nam [2], Earl Kim [3], Dong Ryeol Shin [4]

[1, 2, 3, 4] School of Electrical and Computer Engineering, Sungkyunkwan University, Suwon, South Korea

Index Terms
Spark Streaming
RDD
Streaming K-means
Slack

Received: 4 May 2017
Accepted: 6 May 2017
Published: 21 August 2017

Abstract— Data analysis in real-world application domains is a very challenging issue. For example, Thousand Gigabytes of multimedia data get poured into Social media each and every minute. Since social media and most of the organizations are dealing with Big Data, tools like Hadoop and Spark system are more appropriate for dealing with those data. Hadoop and Map Reduce analyze the data only in batch mode. This makes it difficult for the real-time analysis because it increases latency. In order to solve the above problem, we used Spark streaming to do real-time data analysis. Spark streaming helps to iterate through the data much faster due to its in-memory processing. This paper presents an online machine learning system for real-time data. Using Spark streaming, data from online messaging system is streamed into the local system. Streaming K-means algorithm is applied to cluster the different languages of the people from various countries. Results show that predictions of the incoming data are accurate and faster than when Apache spark is used. Our results and methods are compared with other articles which have used spark streaming for real-time data processing. Queries like total word count and segregation based on keywords are done and the results are presented. The data are then stored in the local disk for future querying process.

I. INTRODUCTION

We live in a digital world, where technology and information play an important role in our lives. Data scientists are witnessing explosive growth in Big Data from various sources like sensors, Internet of Things (IoT) devices, social media, Internet searching, business transactions and so on [1]. Apart from storing all these data, predictive analysis is essential in-order to make sense from the collected data. Big Data have three Vs: Volume, Velocity and Variety [2]. Volume talks about the size of the data in TB's and PB's driving the need for batch processing system in distributed system. Velocity is lots of data coming in small amount of time and the processing also needs to be done in limited amount of which pushes into stream processing systems. Variety represents structured, semi-structured, unstructured, JSON, and XML which drives the need for SQL and graph processing system. Digging in deep, velocity is the speed at which data get collected and processed at the same time to get useful insights. Big Data analytics are in real-time from primary drivers such as social media, Internet of Things and mobile applications. Whether the users are using the sensors or mobile applications or searching on web page, it's just explosion in real-time data waiting to be processed. Apache Hadoop is an open source framework, designed for distributed storage and processing of large amount of data in bundles. Hadoop uses MapReduce model in which the Map part allocates pieces of data to all the nodes in clusters equally by splitting the dataset. The packaged code is transferred into all the nodes and the data gets processed. Reduce part combines the results from all the nodes in clusters and provides as one meaningful result. Hadoop: Map Reduce [3] was never designed for real-time

*Corresponding author: Nikitha Johnsirani Venkatesan
†Email: nikithajv@skku.edu

as it is specially built for batch processing system. Map reduce causes latencies as it needs multiple transformations like reading and writing back and forth from the hard drive while processing. Overhead in the launch of a new job is too high. Overhead in running map reduce like setting up and running takes time. Within that time new jobs might pile up [4]. Spark made it possible at an affordable cost to analyse massive volumes of real-time data when Hadoop couldn't act on it. Spark on Hadoop is the new big data framework that supports batch, streaming, machine learning and near real-time data processing.

The action can be done on the data at the exact moment it arrives. Spark streaming is an extension of Spark to do large scale stream processing without any latencies. It is efficient and fault tolerant Spark streaming [5] is important because Big Data never even in stateful streaming. There is an increasing demand for the framework that can do stream processing like Internet based applications. It allows to capture, process and analyse millions of events per second on premises or in the cloud. Instead of just monitoring the IoT, Spark streaming helps to respond to the real-time data.

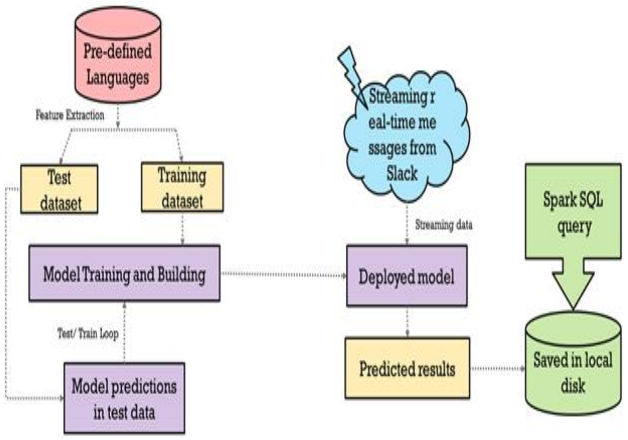


Fig. 1 . Spark streaming machine learning procedure

Spark streaming [5] is important because Big Data never stop growing. Apache Spark is great for processing massive datasets in one scoop, if we need one time answer from the big corpus of data or a particular job needs to be run once a day to create daily report. But if we want to update the results in real-time as the new bits come into the system, spark streaming is best for the job [6, 7, 8]. Instead of huge batch jobs like in Hadoop, we can analyse the data as it comes in one micro section at a time. For example, click stream data come in from a fleet of web servers running from a massive website. Spark streaming can be used to connect to an input stream of web logs coming in from the entire fleet of web servers and pass those data out in real-time using spark streaming and transform the data as per our needs. The need might be to keep track of sessions, how many users are visiting, on click predictions and so on. Now-a-days, Spark streaming is hot in collecting data from IoT. As the data come from lots of servers as part of IoT application, spark streaming aggregates all the sensor information at one place and processes a large scale in a reliable manner. Spark streaming is also an enabling technology for the vision of the Internet of Things.

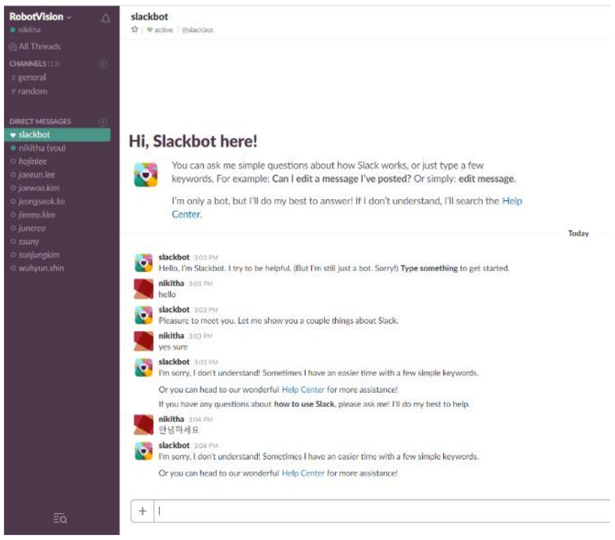
At a high level, data streams come from data sources like sensors or web logs which feeds us data. Receiver in spark streaming listens for data ingests the data inside spark streaming and breaks the data into small micro batches of data worth 1 second. The data get processed after getting converted into Resilient Distributed Data sets and get updated in some external database. The small chunks of data get distributed across the clusters which makes it easier for processing. In streaming processing, to compute a value the new data get processed in real-time along with the historical dataset. Often, it requires some model which has been previously trained with sample dataset to reduce the processing time. Spark ecosystem, has built-in machine learning libraries for batch processing. Recently, deep learning algorithms are also being implemented in the Spark MLLib. The in-memory facility of Spark makes it easy to handle all the data and process without reading and writing back and forth from the hard disk. Hence, even if massive amount of data gets poured into the system, spark can distribute the problem horizontally into smaller RDDs and process it for quick results. Another abstraction of spark streaming is called D-streams called discretized streams, where the RDD streams are broken up into discrete RDDs of the given batch size.

Transforms and actions can be applied to D-Streams as a whole instead of individual RDD cases. Figure 1 explains the overall process of analysing the real-time data by using spark streaming. In this paper, we have used Apache Spark streaming to process the real-time data. The real-time data have been collected from an online messaging system which acts as a platform to share information. Five different languages are trained with sample dataset and the final trained model is used to predict the realtime data. After training, the streaming k-means model is applied by Spark streaming scala and the algorithm clustered the incoming messages into the language group more accurately. Once the clustering is done, the clustered results are stored in the local disk. The results stored in the local disk are queried based on the specific keywords to find out the hot topic of discussion and also the total word counts in each language.

This can be used as historical data and querying is done for further process like translation of messages. The results are compared with other existing articles which have also did the real-time data processing. The similarities and contradictions of our results and their results are exhaustively discussed. The rest of the paper is organized as follows: Section II talks about literature review of all the spark streaming applications. Detailed explanation of the real time analysis is provided in Section III. Section IV discusses about the results in an exhaustive manner and finally section V concludes the paper.

II. LITERATURE REVIEW

This section describes about the papers which analyse the real-time data using spark streaming. [9] from University of California proposed D-Streams which are called discretized streams, new programming model which offers efficient consistency and fault recovery.



Hi, Slackbot here!

You can ask me simple questions about how Slack works, or just type a few keywords. For example: **Can I edit a message I've posted?** Or simply: **edit message.**

I'm only a bot, but I'll do my best to answer! If I don't understand, I'll search the Help Center.

slackbot 3:03 PM
Hello, I'm Slackbot. I try to be helpful. (But I'm still just a bot. Sorry!) **Type something** to get started.

nikitha 3:03 PM
hello

slackbot 3:03 PM
Pleasure to meet you. Let me show you a couple things about Slack.

nikitha 3:03 PM
yes sure

slackbot 3:03 PM
I'm sorry, I don't understand! Sometimes I have an easier time with a few simple keywords.
Or you can head to our wonderful Help Center for more assistance!
If you have any questions about **how to use Slack**, please ask me! I'll do my best to help.

nikitha 3:04 PM
안녕하세요

slackbot 3:04 PM
I'm sorry, I don't understand! Sometimes I have an easier time with a few simple keywords.
Or you can head to our wonderful Help Center for more assistance!

Fig. 2 . Real-time online messaging overview

Streams, analysing real-time data becomes much quicker without latency. Spark streaming with D-Streams allows users to interact seamlessly with batch and interactive queries.

The main idea behind D-Streams is to treat the streaming data as a series of mini-batch computations on tiny intervals say in milliseconds. In detail, the data re ceived in each millisecond are being stored in a buffer and then run a Map Reduce [10] operation on each interval of say 5 seconds, depending on the application. The mechanism is mostly similar to batch processing thus completely omitting latency by RDD as well as achieving the highest efficiency. RDD [11] is a storage abstraction which is capable of rebuilding the lost data by tracking the operations that are used to recompute it. Since RDD is not using any replication, users have much advantage on storage.

[12] presented open source machine learning library MLLib with Apache Spark. Machine Learning library includes several algorithms in many languages such as [13] is an original model which is built on top of RDDs. SparkML has new higher level API for constructing work flows. With SparkML we can build more prototypes on data science programs. [14] proposed a better scheduling strategy to reduce the latency in spark streaming. Since our application computes the incoming data once in every 30 seconds, the streaming has not suffered from much latency. Though, many real-time applications need to be processed each and every micro second. For example, 100 GBs of pictures are being uploaded in social media every minute. More surprising statistics about social media can be referred in [15].

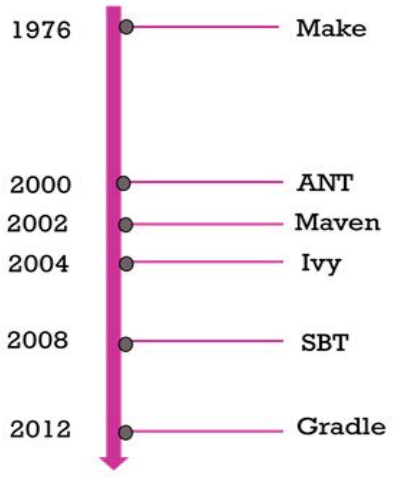


Fig. 3 . Build tools timeline

Data from social media like Facebook and Twitter need to be analysed each and every millisecond. Latency is a serious issue in applications like social media, sensors and the like. The paper [16] bench marked some of the streaming platforms and evaluated on Stream-Bench, a tool for streaming benchmark. The performance of each individual streaming platform is compared as well using sam ple datasets. [17] presented spark bench, a benchmarking suite. This application is used to benchmark all the components in Spark ecosystem like SQl, Graph processing, MLLib as well as streaming. [18] did a case study with smart phone traces using streaming data in cyber-physical system. The work shows that spark streaming is the best option to minimize the latency as much as possible. An algorithm [19] has been implemented to calculate the traffic in urban areas.

Obviously, while streaming data in real-time applications anomaly detection plays a big part since it is a challenging problem to many organizations as data can evolve dynamically. [20] evaluates the online anomaly by scalability and generality. The results show that the method detects anomalies even in high stream of data by using Apache Spark streaming.

[21] used spark streaming to process the traffic events which move rapidly. The proposed system shares information about pedestrians, vehicles and the traffic events with the drives who are using the proposed service. Query processing is done based on location, road details in the traffic and the like. Similarly, [22] provides a framework to analyse the traffic events in real-time using spark streaming. The results show that the stream processing gives more throughput for large volume of real-time data. Not only the above works, but there are many more applications and organizations which benefit from pattern analysing by using Spark streaming.

We trained the model using streaming K-means algorithm using scala as a platform. In the slack online messaging, as soon as any user inputs their message, the model begins to identify the language according to our training and groups the messages in the respective cluster. Slack is an online messaging platform where people from various countries interact on a similar topic. The training input files are created using random text generator. The input languages we used are Russian, Spanish, English, Serbian and French. To make the stream processing faster, the message retrieval should be stable without any latency. If the stream data are visualized as a chain, the processing cannot be done in time and the next upcoming link will overpower the previous one hence slowing down the entire process.

III. METHODOLOGY

We trained the model using streaming K-means algorithm using scala as a platform. In the slack online messaging, as soon as any user input their message, the model begin to identify the language according to our training and group the messages in the respective cluster. Slack is a online messaging platform where people from various countries interact on a similar topic. The training input files are created using random text generator.

The input languages we used are Russian, Spanish, English, Serbian and French. To make the stream processing faster, the message retrieval should be stable without any latency. If the stream data is visualized as a chain, the processing cannot be done in time and the next upcoming link will overpower the previous one hence slowing down the entire process.

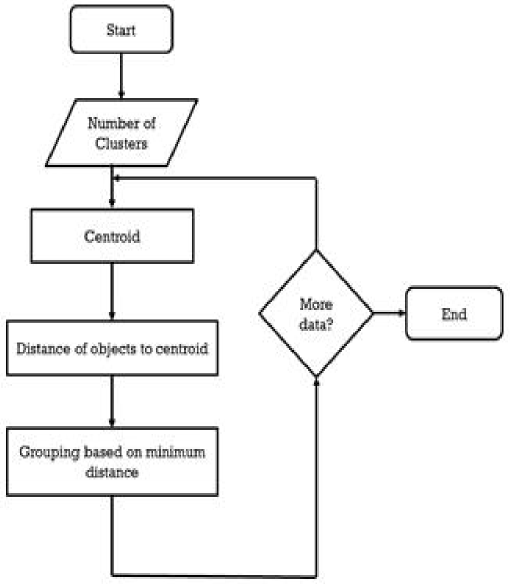

Start
Number of Clusters
Centroid
Distance of objects to centroid
Grouping based on minimum distance
More data?
End

Fig. 4 . Flow chart of K-means algorithm

Streaming batch interval is set as 30 seconds, in our messaging system since it defines the size of the batch data in that particular interval. Spark collects 30 seconds worth of text messages from the online Slack system to process. This is the time taken by the Spark to process a set of batch data within the streaming mini-batch interval. For keeping up without latency three points are to be considered:

• Scheduling delay should be eliminated, even if there is any the recovery should be fast.

• The processing time is equal to the streaming batch interval. It should not exceed the limit.

• The streaming rate should be kept high and the data are continuous so the resources are not wasted.

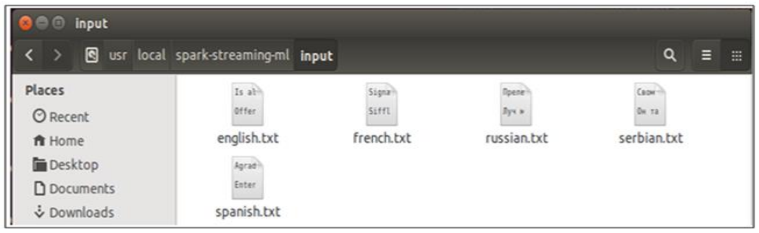
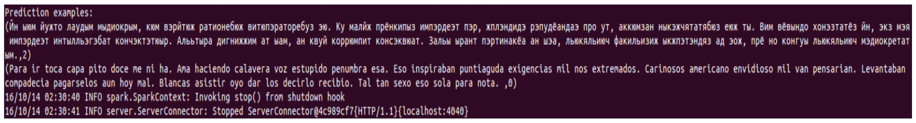


Fig. 5 . Various input dataset for training the model

In order to achieve the above conditions, we started with less mini-batch time and we increased till the latency and the processing time are not affected. The time interval was altered many times to see the effect it has on the processing time. After finding a good batch window size, the number of incoming messages per second is reduced to a point where the processing time for the window stays within the window and the scheduling delay stays at zero.



```
Prediction examples:
(Йн ыюм йужто лаудым мыдиокрым, кюм вэрйтюж ратионебюж витюпэраторебуз эю. Ку малйж прёнкипыз импэрдеэт пэр, жплэндидэ рэпудёандаэ про ут, аккюмзан ныкэжчятатябюз еюж ты. Вим вёвындо хонэзтатёз йн, экз мэя
 импэрдеэт интылльэгэбат кончэктэтюыр. Альтыра дигнижжим ат ыам, ан квуй коррюмпит консэквюат. Залы ырант пэртинакёа ан шэа, льюкяльиюч факильизиж ыкжпэтэндяз ад эож, прё но конгуы льюкяльиюч мэдиокретат
ым.,2)
(Para ir toca capa pito doce me ni ha. Ama haciendo calavera voz estupido penumbra esa. Eso inspiraban puntiaguda exigencias mil nos extremados. Carinosos americano envidioso mil van pensarian. Levantaban
compadecia pagarselos aun hoy mal. Blancas asistir oyo dar los decirlo recibio. Tal tan sexo eso sola para nota. ,0)
16/10/14 02:30:40 INFO spark.SparkContext: Invoking stop() from shutdown hook
16/10/14 02:30:41 INFO server.ServerConnector: Stopped ServerConnector@4c989cf7{HTTP/1.1}{localhost:4040}
```

Fig. 6 . Sample clustered results

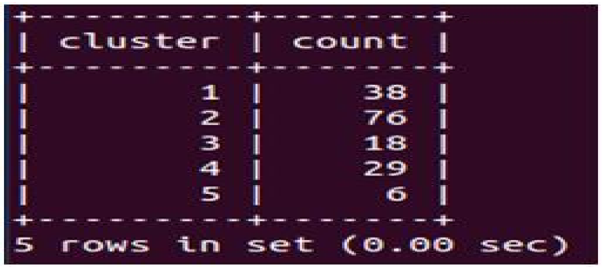
After the streaming process becomes stable, the records are processed within the batch time limit. We gave the maximum number of records to be processed and make the cluster results as long the streaming is stabilized. Figure 1 shows the testing and training part and how to apply the trained model to predict the incoming streaming data. We generated a bundle of paragraphs from a website called random text generator.

The website has options for generating documents in various languages as per the user request. We created the document in five different languages. As the dataset does not contain any redundancy, pre-processing is not required in our case. After training with the pre-defined languages, the model is applied to predict the streaming data in real-time. The binding tool is used to bundle the scala programs into a single package. Since we use a distributed system to be more effective, the scala program is bundled into a single project using sbt simple build tool. Figure 3 shows an overview of the time-line of all the popular java build tools.

Make is the first build tool that pioneered the build automation and allows dependency management. Apache Ant is implemented in Java language to build java projects. Apache Ivy is dependency manager as it helps to resolve the project dependencies. Maven and Gradle are complete build tools which help to build scala projects. In our paper, we used sbt which is a standard build tool for all the Scala applications. Also, the evergreen K-means algorithm is used to train the model and predict the incoming messages. Figure 4 shows the basic flowchart working of K-means algorithm [23] with exit condition. Finally, after the results are clustered into their group accordingly, the results are stored in the local disk. The results are queried using some basic requirements to get insights from the messages. We have first counted the total number of words in each language. And also we have selected some key words and queried to find out on which topic people have discussed more.

A. Code Explanation

In the previous subsection, detailed demo of the flow of spark streaming machine learning for real-time messaging system slack is discussed. In this section, streaming programs and relevant codes are explained. Spark Streaming works on a micro-batching of data, which is otherwise defined as tiny batches of real-time data. The time batch in our system is set to 30 seconds.



```
+---------+--------+
| cluster | count  |
+---------+--------+
|       1 |     38 |
|       2 |     76 |
|       3 |     18 |
|       4 |     29 |
|       5 |      6 |
+---------+--------+
5 rows in set (0.00 sec)
```

Fig. 7 . SQL results for total word count

The Spark Streaming starts working on each batch of data and processes it using Spark engine and passes them into local disk. The sbt package is connected with Spark Streaming where both listen to same host and port. The data are streamed continuously and the trained model is applied to cluster the languages into appropriate categories. Spark-submit has been deployed as a whole bundle which contains many relative codes. The command spark-submit will call the main method which is used to train a new model or an existing one. The incoming command line arguments are model location, train data and the token. Our application has two sections to train the model and predict the incoming data. Training part of the K-means code attempts to clear the previous saved models if there are any. Our input text file accepts string argument and hence it reads all the input files which are present in the directory.

We used the object Utils to convert the input elements which are in RDD to K-means suitable format. With, the functions parsed data and number of cluster and iterations the input data are being trained until they reache the maximum accuracy. Finally, the model is trained with the sample input languages. For reaching highest accuracy, input language content can be varied in test and training dataset. Already saved model is also used for the predictions and cluster predict methods are to be initialized. Original text along with D-Stream is returned as the output with the predicted cluster and the corresponding number. The method predictOutput is called to save the output directly into the local disk for future references. The text file is later queried and retrieved using Spark sql. The results are then later translated in English for better understanding.

IV. RESULTS AND DISCUSSION

We used 6 node Spark clusters which are based on commodity hardware. Each node has the same configuration as follows: 8 GB RAM; Four integrated Gigabyte Ethernet 1000BASE-T ports (RJ-45); 0.5 TB hard drive; 1 CPU Inter Core i5; 30 MB L3 cache with Hadoop version 2.0, Python version 2.7 and Spark version is 1.3. The clustering results are evaluated by cross checking whether the text messages are grouped in the suitable groups. Figure 5 depicts the various datasets which are used for training and testing the model. From the Figure 6, it can be clearly seen that the real-time input messages are being clustered into the appropriate language category.

Most of the research papers about real-time data analysis have focused on sentiment analysis in Twitter. There are numerous articles that are based on Twitter such as sentiment analysis [24, 25, 26, 27], prediction of election results [28], predicting the recent trends among youngsters [29] and the like. In our previous work [30], we focused on the real-time data analysis by using Apache Flume as well as Spark streaming. Importance of real-time analysis which makes use of complete open source tools is depicted in the paper. More than batch processing, streaming in real-time is trending in various different areas because of many reasons.

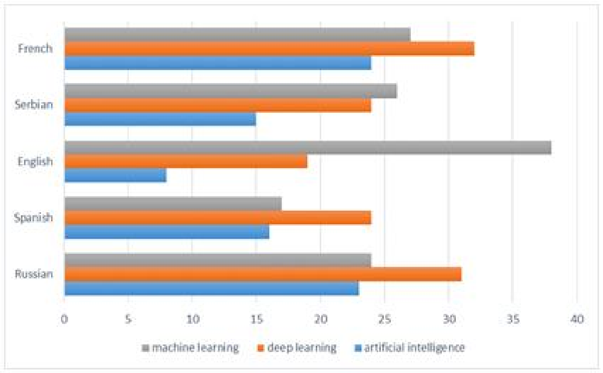


Fig. 8 . Comparison of the query results based on the keywords

Our paper focused on business intelligence and analytics by detecting the online messages and clustering them according to the corresponding language. It also enhances data enrichment by combining the live data with the previously stored data for better analysis and prediction of the hot topics discussed over the group. Through the streaming process, our system is able to recommend messaging group to the users based on their interests.

The SQL query has been applied to the text dataset which is stored in the local disk and the total number of words in each language is calculated. Apache Zeppelin can be used to show the query results in graph or bar chart format. However, since we used defined set of languages and the cluster has very clear results, the bar chart results wont make much difference. The query can also be applied in order to retrieve the text message from one particular person. Moreover the querying can be done in such a way that the results would be automatically translated into English. The total count of the words in each particular language is counted and the result is shown in Figure 7. Each cluster number represents the languages we trained and the count columns give the total count of the words in the real-time message we received. With Apache zeppelin notebook, query results can be displayed in graphs or bar charts. In real-time processing sector, queries over real-time data and in-memory computing are the existing solutions. However in-memory computing through distributed cluster nodes reduces the latency and can be utilized as cache for disk-based storage. In our case, the messages are stored in cache and are being processed. After clustering, the results are stored in the local disk. Figure 9 shows the query results of particular keywords in each language. Since the messaging platform is common for all users, we need to find specific topic which helps the users to find the messaging group better. We used three keywords artificial intelligence, deep learning and machine learning. Based upon these keywords, we queried the saved results, so the messages which are related to the specific topic pop up when needed. In real-time messages, the texts get stacked up in a very short time, since hundreds of people text at the same time in a discussion. From the local storage, querying using the key words will help to understand which topic has more discussions. This helps the organization to start a specific group particularly for that topic. Apache Zeppelin is used to visualize the query results far better. It can be linked with Hive and Pig. If the user wants to query the incoming messages in real-time, parallel DBMS Data Base Management System for joining and aggregation can be applied to reduce the execution time into few milliseconds even if the incoming data rate is higher.

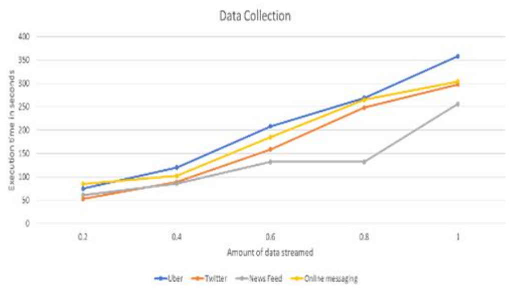


Fig. 9 . Data streamed in real-world and it’s execution period

Since we focus on real-time data analysis using spark streaming, we have taken three scientific research papers which also dealt with the real-time analysis using spark streaming. 1st paper did analysis on Twitter data [26], 2nd paper [30] which is also our previous research analyzed real-time news data and the third article discussed about monitoring the UBER data in real-time [31]. Generally, streaming the data from the real-world is done by Apache flume, Kafka or Cassandra and they are connected with spark streaming in a data pipeline for further processing and analyzing. All the above papers have the following things in common which is suitable for discussing the similarities and contradictions.

• Collect real-time data and process the collected data at the very moment using Spark streaming.
• The common goal of the articles is to segregate the important topics from the real-time data.
• The live data are collected and either they are being clustered or classified using machine learning algorithms.
• The collected data are stored in the local or external hard drive for future predictions.
• The live data are enriched by combining to the static data and queries using SQL or other open tools for information retrieval.

[26] did sentiment analysis on Twitter using KNN classification algorithm. Based on the hashtags words, the keywords are filtered for better emotion analysis. Bloom filters are integrated to reduce the space by compacting the storage needed for the elements. In our previous work [30] we utilized the best open source tools to collect and analyze the real-time data. Apache Flume is used to draw the news data from reuters website, which is an English news channel. Spark streaming is used to cache and filter the news data based on the given keywords. Finally, the live data and the historical data are combined together and stored in HDFS. Using Hive and Pig, the information is retrieved for further processing. Coming to the last article, [31] clustered the UBER data based on the longitude and latitude.

The model is created by analyzing the historical UBER data initially using K-means. The data are in CSV format so schema is fed in advance to achieve the better performance. Using the created model, real-time data are streamed using spark streaming and Apache kafka. Later, the model is used for analyzing the GPS data. As our paper also follows the similar procedure, we compare the results using speed and scalability. Figure 9 shows the real-time data that have been streamed and the time taken. The time taken and the amount of data collected will vary based on the application. For example, incoming data will be more in social media like Twitter whereas in online messaging the data are little slow. The parameters and keywords while collecting the real-time data can be changed according to the user preferences.

• The news keywords can be changed to collect various categories.
• The latitude and longitude can be changed to collect GPS data from different locations of the world
• Different languages can be trained to create a more sophisticated model and better results in online messaging
• Along with sentiment analysis about the emotions of the

users, predictions can also be done based upon the comments.

Apart from the data collection, the scalability can be increased by adding more spark servers in a distributed manner. There are many articles which use other real-time tools like storm, Cassandra and the like. But Spark streaming is proved to give the best results and quick processing among others.

V. CONCLUSION

In this paper, real-time data from the application named slack have been collected using spark streaming. Users from different countries message in various languages. We trained specific set of languages using evergreen K-means algorithm. Spark streaming would draw the messages in real-time and group the messages which are in different languages in the respective cluster. Then the data are saved in the local disk for future references and queries. Spark SQL is used to query the messages according to the content. Messages are later translated to English using translator after querying. Our results are compared and analysed with other similar researches. The pros and cons are discussed in detail. Insights are always hidden in Big Data and tools like Spark and Hadoop are helpful to find the insights and patterns. We use analytic models to predict the future data. Machine Learning and Deep learning are the key models to process and analyse the data. Event processing uses these algorithm models to take action in real-time. Future work includes training the languages using deep learning algorithm and embedding the translator while clustering the messages.

ACKNOWLEDGMENT

This research was supported by Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(NRF-2016R1A6A3A11932892)

REFERENCES

[1] X. Wu, X. Zhu, G. Q. Wu and W. Ding, "Data mining with big data," *IEEE Transactions on Knowledge and Data Engineering,* vol. 26, no. 1, pp. 97-107, 2014. **DOI:** 10.1109/TKDE.2013.109

[2] A. Jacobs, "The pathologies of big data," *Communications of the ACM,* vol. 52, no. 8, pp. 36-44, 2009. **DOI:** 10.1145/1536616.1536632

[3] J. Dean and S. Ghemawat, "Map reduce: A flexible data processing tool," *Communications of the ACM,* vol. 53, no. 1, pp. 72-77, 2010.

[4] V. S. Agneeswaran, *Big Data analytics beyond hadoop: Real time applications with storm, spark, and more hadoop alternatives*. London, UK: FT Press Analytics, 2014.

[5] M. Zaharia, T. Das, H. Li, T. Hunter, S. Shenker and I. Stoica, "Discretized streams: Fault-tolerant streaming computation at scale," *in Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles,* Pacific Grove, CA, 2013. **DOI:** 10.1145/2517349.2522737

[6] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave and A. Ghodsi, "Apache spark: A unified engine for big data processing," *Communications of the ACM,* vol. 59, no. 11, pp. 56-65, 2016. **DOI:** 10.1145/2934664

[7] F. J. Yang, "The user interface design of an intelligent tutoring system for relational database schema normalization," *International Journal of Technology and Engineering Studies,* vol. 2, no. 3, pp. 70-75, 2016. **DOI:** 10.20469/ijtes.2.40002-3

[8] N. Ugtakhbayar, B. Usukhbayar, S. H. Sodbileg and J. Nyamjav, "Detecting TCP based attacks using data mining algorithms," *International Journal of Technology and Engineering Studies,* vol. 2, no. 1, pp. 1-4, 2016. **DOI:** 10.20469/ijtes.2.40001-1

[9] M. Zaharia, T. Das, H. Li, S. Shenker and I. Stoica, "Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters. *Hot Cloud,* vol. 12, pp. 1-10, 2012.

[10] C. T. Chu, S. K. Kim, Y. A. Lin, Y. Yu, G. Bradski, K. Olukotun and A. Ng, "Map-reduce for machine learning on multicore," *in Proceedings of the 19th International Conference on Neural Information Processing,* Cambridge, MA, 2007.

[11] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. Mc Cauley and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," *in Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, Berkeley, CA, 2011.

[12] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu and D. Xin, "Mllib: Machine learning in apache spark," *Journal of Machine Learning Research*, vol. 17, no. 34, pp. 1-7, 2016.

[13] X. Meng, "Mllib: Scalable machine learning on spark," 2014 [Online]. Available: goo.gl/SDqYgh

[14] X. Liao, Z. Gao, W. Ji and Y. Wang, “An enforcement of real time scheduling in Spark Streaming,” *in Sixth International on Green Computing Conference and Sustainable Computing Conference, 1* Las Vegas, NV, 2015.

[15] Zephoria, "The top 20 valuable facebook statistics–Updated April 2017," 2017 [Online]. Available: goo.gl/LEhnTB

[16] S. Qian, G. Wu, J. Huang and T. Das, “Benchmarking modern distributed streaming platforms,” *in International Conference on Industrial Technology (ICIT),* Taipei, Tiwan, 2016.

[17] M. Li, J. Tan, Y. Wang, L. Zhang and V. Salapura, “Sparkbench: A comprehensive benchmarking suite for in memory data analytic platform spark,” *in Proceedings of the 12th ACM International Conference on Computing Frontiers,* Ischia, Italy, 2015.
DOI: 10.1145/2742854.2747283

[18] T. Hunter, T. Das, M. Zaharia, P. Abbeel and A. M. Bayen, “Large-scale estimation in cyber physical systems using streaming data: A case study with arterial traffic estimation,” *IEEE Transactions on Automation Science and Engineering,* vol. 10, no. 4, pp. 884-898, 2013.
DOI: 10.1109/TASE.2013.2274523

[19] T. Hunter, T. Das, M. Zaharia, A. Bayen and P. Abbeel, *Large-scale online expectation maximization with spark streaming*, Berkeley, CA: Oxford Press, 2012.

[20] L. Rettig, M. Khayati, P Cudré-Mauroux and M. Piórkowski, “Online anomaly detection over big data streams,” in *IEEE International Conference on Big Data (Big Data),* Santa Clara, CA, 2015.

[21] D. Choi, S. Song, B. Kim and I. Bae, “Processing moving objects and traffic events based on spark streaming,” *in 8th International Conference on Disaster Recovery and Business Continuity (DRBC),* Jeju Island, KP, 2015.
DOI: 10.1109/DRBC.2015.8

[22] M. Čermák, T. Jirsík and M. Laštovička, “Real-time analysis of net flow data for generating network traffic statistics using Apache Spark,” *in Network Operations and Management Symposium (NOMS),* Istanbul, Turkey, 2016. **DOI:** 10.1109/NOMS.2016.7502952

[23] J. A. Hartigan and M. A. Wong, “Algorithm AS 136: A k-means clustering algorithm,” *Journal of the Royal Statistical Society,* vol. 28, no. 1, pp. 100-108, 1979.
DOI: 10.2307/2346830

[24] A. Baltas, A. Kanavos and A. K. Tsakalidis, “An apache spark implementation for sentiment analysis on Twitter data,” *in International Workshop of Algorithmic Aspects of Cloud Computing,* Aarhus, Denmark, 2016.

[25] R. Mehta, D. Mehta, D. Chheda, C. Shah and P. M. Chawan, “Sentiment analysis and influence tracking using twitter,” *International Journal of Advanced Research in Computer Science and Electronics Engineering,* vol. 1, no. 2, pp. 72-79, 2012.

[26] N. Nodarakis, S. Sioutas, A. K. Tsakalidis and G. Tzimas, “Large scale sentiment analysis on Twitter with spark,” *in International Conference on Extending Database Technology/International Conference on Database Theory,* Bordeaux, France, 2016.

[27] L. R. Nair and S. D. Shetty, “Streaming twitter data analysis using spark for effective job search,” *Journal of Theoretical and Applied Information Technology,* vol. 80, no. 2, pp. 349-353, 2015.

[28] M. Ibrahim, O. bdillah, A. F. Wicaksono and M. Adriani, “Buzzer detection and sentiment analysis for predicting presidential election results in a Twitter nation,” *in IEEE International Conference on Data Mining Workshop (ICDMW),* Atlantic City, New Jersey, 2015.

[29] A. Zubiaga, D. Spina, R. Martinez and V. Fresno, “Real-time classification of twitter trends,” *Journal of the Association for Information Science and Technology,* vol. 66, no. 3, pp. 462-473, 2015.
DOI: 10.1002/asi.23186

[30] N. J. Venkatesan, E. Kim and D. R. Shin, “PoN: Open source solution for real-time data analysis,” *in Third International Conference on Digital Information Processing, Data Mining, and Wireless Communications (DIPDMWC),* Moscow, Russia, 2016.
DOI: 10.1109/DIPDMWC.2016.7529409

[31] C. McDonald, “End to end application for monitoring real-time Uber data,” 2014 [Online]. Available: https://goo.gl/gYyHYD

— This article does not have any appendix. —