## **Introduction**

Studies have forecasted that there will be trillions of digital entities, billions of connected devices and millions of microservices. When these entities interact purposefully, the output is the tremendous amount of multi-structured data. When such a humungous number of entities getting connected to the Internet, the result is the exponential growth in the size of the Internet. The scope and structure of how the Internet will evolve in the coming years is simply phenomenal. According to a survey from OurWorldInData (Roser et al., 1988), around two-thirds of the world’s population have the access to the Internet. There is a fruitful and fulsome convergence of information and communication technologies with operational technologies. These distinct improvements and innovations have led to the accumulation of a massive quantity of value-adding data.

However, for performing sentiment analysis, programmatically gathering, storing, analyzing and extracting actionable insights through a host of integrated data analytics platforms and artificial intelligence (AI) algorithms is inefficient and expensive. Though the analytical technologies and tools have matured and stabilized, the conventional implementation works out to be inefficient and can be optimized much further.

Sentiment analysis has been a notable research topic over the last few years, but the focus has always been on improving the model rather than the overall system architecture. Research by D. Sculley and his team at Google (Sculley et al., 2015) shows that only a small fraction of real-world machine learning (ML) systems contains ML code. The remaining code comprises of configuration, data collection and verification, machine resource management (Shao et al., 2009), process management, feature extraction and engineering, governance and monitoring logic. So, there is a need for disaggregation to simplify and streamline the analytical process in a precise and concise manner. Thus, besides employing a new approach, the execution has to be substantially automated. The new arrival of serverless computing is being embraced in order to ensure deeper and decisive automation. Figure 1 vividly illustrates the scope for improvement with an enhancement in the system architecture.



Figure 1. Only a small portion of real-world machine learning (ML) systems are composed of ML code, as shown by the black box in the middle

From this, it is implied that the optimization in the ML code alone does not result in a large-scale improvement in the overall functioning of a real-world ML-based system. In this paper, we propose a new optimized sentiment analysis system which fully utilizes the power of serverless computing and cloud resources to deploy a sentiment analysis engine that is capable of performing opinion mining tasks in a highly optimized manner and consumes fewer computing resources. These improvements have laid down a stimulating foundation for other researchers to think ahead, as serverless deployments have brought down the initial and recurring costs substantially (Rajan, 2018), in comparison to traditional deployments.

**What is Sentiment Analysis**

This is alternatively called as opinion mining. This is primarily achieved using the proven and potential natural language processing (NLP) techniques, which is meticulously used to determine whether the incoming opinion data is positive, negative or neutral. As we know, natural language processing (NLP) is a prominent module of Artificial intelligence (AI). For enabling computers and communicators to gain human-like speech synthesis and recognition capabilities, NLP techniques and tools are methodically leveraged. As sentiments, opinions and viewpoints are being predominantly expressed through texts, the concept of text mining or analytics is becoming popular. Text analytics is done through a host of promising and potential NLP methods. This analytics process emits out actionable insights, which can be gleaned and looped back to machines and devices to have a logically correct conversation with others in the vicinity. Also, the NLP competency makes it easy and unambiguous for machines and people to interact logically.

There are case studies clearly articulating that sentiment analysis is often performed on textual data to help businesses minutely monitor brand value and product sentiments through sustained customer feedbacks (Sivarajah et al., 2020). Customer satisfaction is the main motive behind this initiative for worldwide enterprises. For example, sentiment analysis can help to automatically analyse thousands of product reviews. The knowledge discovered empowers decision-makers and product designers to do a course correction and to come out with updated/upgraded products through subsequent releases.

It is possible to gauge brand sentiment on social media proactively and pre-emptively. This helps service providers and business organizations to plunge into appropriate actions that ultimately help to pinpoint unsatisfied customers and how they can be approached to soothe them in time. Such proactiveness goes a long way in empowering organizations to retain their customers and to get new customers. Sentiment analysis is extremely important because it helps businesses quickly understand the overall opinions of their customers. By automatically funnelling, slicing and dicing the customer reviews and social media posts and conversations, it is possible to understand what people think. This empowerment enables decision-makers to take faster and corrective decisions well before any irreparable and incorrigible thing happens. However, here are technical challenges for doing sentiment analysis. The world generates a lot of data today (around 3 zettabytes per day). Most of the data generated and collected are unstructured data. People and professionals express their opinions and feelings in the form of reviews, comments, complaints, chats, social media posts, etc. Also, institutions, governments, and corporates come out with scores of documents in the form of blogs, survey articles, research publications, white papers, case studies, invoices, etc. Precisely speaking, it is data-driven insights and insights-driven decisions and deals. But performing in this big data era is beset with several business and technical challenges.

**Creating an efficient sentiment analysis algorithm**

Rule-based systems are popular and predominant. Now with the widespread utilization of AI methods, automatic methods, which do not depend on manually crafted rules, are gaining prominence these days. Automatic approaches are mainly based on machine learning algorithms. Sentiment analysis comes under the supervised learning. A classifier is developed through training, refining and continuous testing of the algorithm. After the classifier id developed when new text data is fed into the classifier, it responds with a category or score which defines is generally for the classes positive, negative or neutral.

Figure 2 enlists the steps to be taken for arriving at competent machine learning (ML) model.



Figure . The Machine Learning (ML) Process for Sentiment Classification

#### **Feature Extraction from Text –** This is the first step for developing a useful ML text classifier. Value-adding features have to be astutely extracted from the input data (text). There are several classical approaches in recent literature in order to assist the important aspect of feature engineering. A widely used approach for this task is bag of words or bag of n-grams with their frequency. New approaches are emerging and evolving in this field. These come in handy for sharply improving the performance of classifiers.

#### **Classification Algorithms -** The classification step usually involves a statistical model like Naïve Bayes, Logistic Regression, Support Vector Machines, or Neural Networks. Naïve Bayes is a family of probabilistic algorithms that uses Bayes Theorem to predict the category of a text. In our system, we have used the Naïve Bayes algorithm as it tends to scale well in comparison to other algorithms.

#### The serverless approach is facilitated through the power of machine learning algorithms and is leveraged in our sentiment analysis system.

## **Literature Review**

In the study by (Krishna et al., 2013), a personal recommendation system was proposed by the author. The design was based on Learning automata and sentiment analysis. Here Local Authority Software Applications (LASA) is used to recommend places adjacent to the present user’s location after the location data is vectorised. However, the security aspects of the recommender systems were not considered

In 2018, (Abdi et al., 2018) suggested a machine learning technique to summarize the opinions of users mentioned in reviews. It involved merging different features into a single feature set for creating a classification model. The performance was measured for four of the main feature selections models that attained the best performance. The performance was measured using seven classifiers which were used in selecting a feature set. This method was implemented in various datasets and the results have concluded that using Information Gain as the feature selector along with SVM for classification optimized the performance.

(Yang et al., 2013), proposes a recommendation system that is based on Bayesian inference for online social networks. Ratings are shared with friends after which conditional probability is used to calculate rating similarity. Their work shows that their proposed Bayesian inference-based recommendation is more accurate than existing trust-based recommendations. This method is also able to solve the problem of having a large size of an element in collaborative filtering. The main demerit of this method is the rating sparseness issue.

(Anto et al., 2016) proposed a method to acquire user feedbacks in conditions when most of the users do not give feedback, by automatically extracting data about a specific product from Twitter. Text classification models are compared out of which Support Vector Machines (SVM) is shown to be the most accurate. The Twitter API is used for data extraction, the main demerit of this method is that accurate product identification in the Twitter data pool is not possible across all domains.

(Afzaal et al., 2019) have created a novel method in utilizing an aspect-based sentiment analysis system that recognized features in text accurately and obtained a great classification accuracy. This was put into practice in a mobile application that assisted tourists in identifying the best hotels in their locality, from the results presented we can conclude that the presented model was effective in both recognition and classification.

(Vashishtha & Susan, 2019) have used a fuzzy rule-based unsupervised model which is in combination with Word Sense disambiguation on a collection of multiple datasets and lexicons. the experiments were done on 9 public Twitter datasets and also utilized 3 sentiment lexicons. The results show that the proposed model gives good results on the utilized dataset.

(Neri et al., 2012). proposed a knowledge mining system which is done on 1000 Facebook posts the proposed system uses web crawlers to mine data and create a dataset to cross-reference, the analysis only monitored two companies Rai and La7.this system however impractical in the real-world application as it is highly impossible to store and analyze a large amount of data being generated when we increase the companies or topics we need to monitor.

The proposed system aims to solve the problems which are being faced by current sentiment analysis systems, and succulently optimizes resource cost for processing large data.

**Understanding Serverless Computing**

Without an iota of doubt, the concept of serverless computing is seen as a paradigm shift in IT operations. The server management is being taken care of by the cloud service provider. There are competent technologies such as containers, functions, Kubernetes, and other enablers and they combine together to create deeper automation in server management. Self-healing, scaling, deployment and management of serverless functions and applications come as a savior for the IT operations team. The real pay per usage concept is fully materialized through the concept of serverless. Automation is continuously insisted across the machine and deep learning (ML/DL) model generation and faster and frequent deployment to eliminate operations teams’ errors and also to significantly reduce the operational and management complexities and costs. We have fully embraced this futuristic and flexible IT approach to come out with a novel solution to bring down system costs through automation. The distinguished aspects of serverless computing are being delineated below in detail.

The **front-end architecture** of serverless computing incorporates a set of tools that invariably focus on the user interface (UI). This is to greatly enhance user experience and to fruitfully mediate between UI designers and back-end developers. We have a variety of input/output devices such as laptops, tablets, smartphones, handhelds and wearables these days for inputting data into backend applications and to view outputs. Increasingly applications are web and mobile-enabled. With web servers are increasingly run on cloud infrastructures, we started to call them cloud applications. Further on, there are web and mobile browsers in most of the client devices. These are generally touted as thin clients. Clients enabled with web applications are called thick clients.

Much of the data processing tasks are being taken to faraway cloud environments in order to reap the originally expressed benefits of the cloud computing paradigm. Very specific actions are being kept in client devices and all kinds of complex and common functionalities are being accomplished through the infinitely powerful cloud platforms and infrastructures.

The **Backend architecture** includes the back-end application running on web servers and the corresponding database running on database servers in faraway cloud environments. Data storage appliances or services also play a very vital role in shaping up tiered or layered applications. With the surge in cloud adoption, the leverage of cloud infrastructure modules and platforms has gone up significantly. The well-known compute machines include bare metal (BM) servers, virtual machines (VMs), and containers. Then there are storage modules being provided as a service by cloud service providers. In most cases, users have to monitor, manage and maintain their infrastructure modules. Now with the sweeping idea of serverless computing, infrastructure management is being insightfully automated and fully delegated to cloud infrastructure experts. The idea is to empower software engineers and developers to concentrate on their specialised skills and not to bother about setting up and sustaining appropriate infrastructures for hosting and running applications.

**Database Systems -** Databases are important for any dynamic and enterprise-scale applications. Databases are stored and managed on database servers. Authorized users are only allowed to ensure the tightest and unbreakable security for data getting persisted in databases. Databases are being accessed by clients via the middle layer (the back-end business application). With the dawn of the big data era, the idea and implementation of NoSQL databases have flourished with the proper nourishment from database product vendors. Compared to SQL databases, which have been doing yeomen service for production-grade applications, NoSQL databases carry some distinct advantages. NoSQL supports BASE (basically available, soft state and eventually consistent). As per the CAP theorem, only two of the three (consistency, availability and partition tolerance) can be availed at the same time. NoSQL, as a primary storage mechanism for big data, is blessed with high availability and partition tolerance. Thereby horizontal scalability is being easily fulfilled by NoSQL systems. There are more than 20 NoSQL database systems in the market. Cassandra, MongoDB, CouchDB, Redis, etc. are some of the renowned NoSQL databases. The new solution articulated in this paper leverages DynamoDB offered by the AWS cloud to store pre-processed text data.

There are other aspects getting associated with the serverless era. The tool ecosystem is continuously growing in order to make the path-breaking serverless phenomenon penetrative, participative and pervasive. Monitoring, measurement and management of serverless applications are important to identify any limitation and to surmount them in an automated manner. The security of serverless functions and applications is very essential for the concept to surge ahead. As the total control of serverless applications is with the cloud service provider, the security of serverless application and data is drawing attention from the security experts and researchers.

As we all know, security is being proclaimed as the barrier to public cloud adoption. Now, with the complete control of running serverless applications is handed over to cloud providers, the security phenomenon is getting extra attention. Cloud services providers are investing their talent, time and treasure in ensuring impenetrable security for the customer, confidential and corporate data. As cyberattacks are on the rise, service providers are focusing on enhanced security through a host of security mechanisms. Serverless security will be a key factor in the days to come.

Finally, there are a number of serverless platforms from the open-source community. Further on, there are several commercial-grade platform vendors building serverless platforms. Serverless platforms are being deployed in public and private cloud environments. All kinds of disclosed automation are being facilitated through competent serverless platforms.

**The System Architecture and the Key Components**

The proposed sentiment analysis system consists of the following components.

1. **Live data sources** such as the Twitter, Reddit or Instagram API and RSS News feeds are used for gathering live data sources. This is done to reduce the database resource cost that would be incurred if a web crawler was utilised to get data from diverse sources. Another advantage of using this approach is that web crawlers can sometimes take days to perform a complete search and includes a lot of noise from many other sites. However, using live data sources will give you similar results within seconds as we are directly integrating with the data from the source you want to analyse.
2. **Analysis Engine** is used to query the relevant information across the data sources and identifies semantic relations in the data. Custom pre-processing algorithms are used as the data is sourced from an aggregate of many data sources, and also, they have to be cleaned and processed differently. Lexical analysis is done. The average polarity and subjectivity are calculated for each result returned. The returned result is then stored with the URL of the data to create a server cache. This entire engine is deployed in a serverless manner so resources will be utilised only when a search call is made
3. **NoSQL cloud database** is used to store the analysed sentiment logs as it will be capable of handling a large volume of data with lower latency and is more scalable when compared with a relational database management system (RDBMS).
4. **Front-End Application** is utilised to call the analysis engine and query the database to show relevant visualisations on the semantic search within a certain configurable timeline. This application can also be used to set a time-based trigger on the analysis engine to perform live public opinion mining.

This entire system is designed such that resources will be used only when required. It does not require constantly running servers, managing very large databases or running web crawlers to search the web to perform sentiment analysis on public opinion. The system architecture is given below. The end-to-end workflow is depicted in the architecture diagram below. The technologies and tools used for the proposed lean and lithe sentiment analysis are vividly indicated in figure 3. We have used DynamoDB of AWS cloud as the NoSQL database. But our system can use any standard NoSQL database.

****

Figure 3. Proposed Sentiment Analysis System deployed on AWS Cloud

### **Live Data Sources**

### There are a variety of live data streams that can be utilised with this system but each of these sources requires a different method of querying data streams and they also return different responses when called. This is the reason why we need to normalise the data after aggregation. Also, different data streams require unique methods of normalisation. These data sources are.

**RSS/ News Feeds** - RSS feeds are specified in XML. That means data must be extracted between the respective XML tag. This can be done using regular expressions or other text extraction libraries. Since articles contain a lot of information, a summary must be created to extract only the necessary key points. This can be done by using simple text summarization algorithms.

**Twitter API -** Twitter data gives us a good insight into public opinion but the quality of the dataset depends on how well we can pre-process the data. For pre-processing, Twitter handles, punctuation and stop words are removed from the tweet as they only add additional noise into the data. Tokenisation and Lemmatization are then performed to extract keywords from the tweet.

**Reddit API -** The Reddit API requires pre-processing steps which are similar to the Twitter API, as the text data has a similar structure. Punctuation and stop words are removed from the post or comment. We then perform Tokenisation and Lemmatization to extract keywords from the data.

### **Analysis Engine**

This component is used to control the overall working of the application. It extracts relevant information from the data using pre-processing algorithms that are customised to the type of incoming data. The ultimate goal of this important component is to create a system that is only active when there is a stream of data related to the semantic search query. This can be achieved either by activating the server based on a fixed time interval or dynamic interval trigger which will be adjusted according to the number of results acquired in the semantic search. The aggregated data is then normalised and stored in a temporary database. This data is then tokenised and a binary sentiment score is calculated for it. Then the calculated sentiment score is then stored in the NoSQL database.



Figure 4. Workflow of the analysis engine

## **NoSQL Cloud Database**

For this application, we have chosen a NoSQL cloud database over a SQL database as it offers more flexibility and has a lower latency. NoSQL databases are horizontally scalable, which means the server load can be reduced by connecting additional servers to the database. This is in comparison to a standard SQL database which is vertically scalable, where for decreasing the server load, the database has to be taken offline and server components like RAM, SSD or CPU need to be upscaled. Along with the added advantage of using a NoSQL database, since the database is deployed on the cloud, the incurred costs can be optimized on a pay as you go billing method.

## **Front-End Application**

The front-end application will be used to query and visualize the extracted data from the database. This application is deployed in a separate serverless module using Django or Flask to prevent constant loading of the analysis engine every time a query is made. The primary function of the front-end application is to serve as a dashboard for the semantic search results which are stored in the database. This also helps in modifying the keywords that are used in the semantic search. The application can also be used to modify the time interval of the trigger used to run the analysis engine.

These four components are the key contributors to the application. Data sources, analytics engine, storage and visualization gel well to support the ideals of the proposed applications. The application is made to run on AWS cloud environments. This application is designed and developed as cloud-agnostic and hence the prickling issue of vendor lock-in is not there. This application can leverage any NoSQL database and hence technology update and upgrade can be accomplished easily and quickly. There are other advantages, which are discussed below.

## **Experimentation Results**

## **Performance Comparison** - To highlight the effectiveness of this new system, we compare the proposed system with two of the most broadly classified methods in the scientific literature (Jabbar et al., 2019), (Jayashreeet & Kulkarni, 2017), (Htet et al., 2019), (Gottipati et al., 2018) with respect to a serverless live sentiment analysis system.

1. The server is always running and the model is always loaded into the memory
2. The server is always running and the model is loaded when data is detected in the stream

The simulated server tests were performed on a system with an intel i7 processor and 16 GBs of memory. The proposed serverless approach test was carried on AWS lambda with 512 MB of RAM. We have used a Naïve Bayes classifier trained on the movie reviews dataset (Maas et al., 2011), to compare its efficiency across these different methods. The read only memory used by this sentiment classifier was calculated to be 48.71Mb. The time taken to load the model was 1.03 seconds and the time taken to make a prediction was 0.0003 seconds

## **Server is always running and the model is always loaded into the memory** - In this type of approach, the model is loaded into the memory even when there is no data input and the server is connected to a live data stream or web crawler. We are comparing this approach to the proposed approach running at an interval of 1 hour in terms of RAM usage per second. For this experimentation, 100000 results are being generated per hour from the semantic search query.



Figure 5. Memory comparison

From the results obtained, we could see that the proposed approach is 92% more memory-efficient in the first hour and the efficiency increases with time when compared with the traditional approach. This efficiency also can be translated to efficiency in memory-dependent components such as electricity consumed, computing power etc.

## **Server is always running and model is loaded when data is detected in the stream** - In this type of approach, the model is loaded only when data is detected from the live stream. This might be more memory-efficient but is a very unscalable approach due to the delay caused by loading the model. We are comparing this approach to the proposed approach running it terms of scalability using the time taken as the number of results increases.



Figure 6. Time comparison

From the results, we can see that initially, the proposed approach timing is almost at par with the existing approach, but when the existing approach processes a large number of results, the time taken to load the model multiple times accumulates a delay which results in an exponential time increase in processing the results

## **Price Comparison**

To test the price optimization that our approach has achieved, we compare it with a server-based approach that runs with an integrated web-crawler. We will be comparing a lambda function with 256 MB of RAM that will run for 60 seconds to process 100,000 results. Running a web crawler requires at least 4 GB of RAM to perform large scale web crawling. The is why we compare our proposed solution in AWS lambda to the AWS EC2 server instances t3.medium, t3.large, t3.xlarge with on-demand pricing. We perform the comparison by calculating the total cost in Lambda and the EC2 instances to process 106 requests (106.106 semantic search results) over a period of a month.

To calculate the cost of the lambda-based approach we have used the formula

Cλ is the total cost for a number of requests

Crλ is the fixed cost per request.

di is the duration of the function (in ms)

C’λ is the cost per second for the serverless function.

N is the total number of requests in a given period.

To calculate the total cost for the EC2 instances, we use:

ECT is the cost function for a given period T for the EC2 instance.

rt is the number of requests to process per second.

rmax is the maximum number of requests an instance can process per second.

CEC2 is the cost per time unit (seconds) for a given cloud instance.

T is the time period for the cost analysis.



Figure 7. Cost Comparison

From the results, we could see that the lambda function cost stays at 0 for the first 9 days of the test. This is because till this point it is under the limits of the AWS free tier. For the other EC2 instances, the cost scales linearly. This shows that even for analyzing large amounts of results (1012) the lambda approach is the cost-effective method.

## **Conclusion**

In this paper, we have presented an implementation of a sentiment analysis system, which is primed for serverless computing. The structure of the system’s architecture and component level optimization was discussed. Two of the most prevalent methods in sentiment analysis systems have been compared with the proposed serverless approach. We have discussed and tested the key areas in which existing approaches can be optimized further. The cost comparison is also shown in order to clearly articulate how this system is able to utilize the distinct advantages of serverless computing. We have also highlighted how the system is able to take advantage of its design to provide exponential optimization in performance and cost.

## **References**

[1] Roser, M., Ritchie, H., & Ortiz-Ospina, E. (2015). Internet. Our World in Data. https://ourworldindata.org/internet

[2] Sivarajah, U., Irani, Z., Gupta, S., & Mahroof, K. (2020). Role of big data and social media analytics for business to business sustainability: A participatory web context. Industrial Marketing Management, 86, 163–179. https://doi.org/10.1016/j.indmarman.2019.04.005

[3] Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J. F., & Dennison, D. (2015). Hidden technical debt in machine learning systems. Advances in Neural Information Processing Systems, 2015-Janua, 2503–2511.

[4] Rajan, R. A. P. (2018). Serverless Architecture - A Revolution in Cloud Computing. 2018 10th International Conference on Advanced Computing, ICoAC 2018, 88–93. https://doi.org/10.1109/ICoAC44903.2018.8939081

[5] Shao, Z., Jin, H., & Li, Y. (2009). Virtual machine resource management for high performance computing applications. Proceedings - 2009 IEEE International Symposium on Parallel and Distributed Processing with Applications, ISPA 2009, 137–144. https://doi.org/10.1109/ISPA.2009.52

[6] Krishna, P. V., Misra, S., Joshi, D., & Obaidat, M. S. (2013). Learning Automata Based Sentiment Analysis for recommender system on cloud. 2013 International Conference on Computer, Information and Telecommunication Systems, CITS 2013. https://doi.org/10.1109/CITS.2013.6705715

[7] Abdi, A., Shamsuddin, S. M., Hasan, S., & Piran, J. (2018). Machine learning-based multi-documents sentiment-oriented summarization using linguistic treatment. Expert Systems with Applications, 109, 66–85. https://doi.org/10.1016/j.eswa.2018.05.010

[8] Yang, X., Guo, Y., & Liu, Y. (2013). Bayesian-inference-based recommendation in online social networks. IEEE Transactions on Parallel and Distributed Systems, 24(4), 642–651. https://doi.org/10.1109/TPDS.2012.192

[9] Anto, M. P., Antony, M., Muhsina, K. M., Johny, N., James, V., & Wilson, A. (2016). Product rating using sentiment analysis. International Conference on Electrical, Electronics, and Optimization Techniques, ICEEOT 2016, 3458–3462. https://doi.org/10.1109/ICEEOT.2016.7755346

[10] Afzaal, M., Usman, M., & Fong, A. (2019). Tourism mobile app with aspect-based sentiment classification framework for tourist reviews. IEEE Transactions on Consumer Electronics, 65(2), 233–242. https://doi.org/10.1109/TCE.2019.2908944

[11] Vashishtha, S., & Susan, S. (2019). Fuzzy rule based unsupervised sentiment analysis from social media posts. Expert Systems with Applications, 138, 112834. https://doi.org/10.1016/j.eswa.2019.112834

[12] Neri, F., Aliprandi, C., Capeci, F., Cuadros, M., & By, T. (2012). Sentiment Analysis on Social Media. https://doi.org/10.1109/ASONAM.2012.164

[13] Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. ACL-HLT 2011 - Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, 1, 142–150. https://www.aclweb.org/anthology/P11-1015

[14] Jabbar, J., Urooj, I., Junsheng, W., & Azeem, N. (2019). Real-time sentiment analysis on E-Commerce application. Proceedings of the 2019 IEEE 16th International Conference on Networking, Sensing and Control, ICNSC 2019, 391–396. https://doi.org/10.1109/ICNSC.2019.8743331

[15] Jayashree, R., & Kulkarni, D. (2017). Recommendation system with sentiment analysis as feedback component. Advances in Intelligent Systems and Computing, 547, 359–367. https://doi.org/10.1007/978-981-10-3325-4\_36

[16] Htet, H., Khaing, S. S., & Myint, Y. Y. (2019). Tweets sentiment analysis for healthcare on big data processing and IoT architecture using maximum entropy classifier. Advances in Intelligent Systems and Computing, 744, 28–38. https://doi.org/10.1007/978-981-13-0869-7\_4

[17] Gottipati, S., Shankararaman, V., & Lin, J. R. (2018). Text analytics approach to extract course improvement suggestions from students’ feedback. Research and Practice in Technology Enhanced Learning, 13(1), 6. https://doi.org/10.1186/s41039-018-0073-0