National College of Ireland

Improving sentiment analysis using containerized microservices approach

Research in Computing
MSc in Cloud Computing

Bharat Goyal
Student ID: x19215860

School of Computing
National College of Ireland

Supervisor: Dr. Rashid Mijumbi

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Bharat Goyal
Student ID:	x19215860
Programme:	MSc in Cloud Computing
Year:	2021
Module:	Research in Computing
Supervisor:	Dr. Rashid Mijumbi
Submission Due Date:	16/12/2021
Project Title:	Improving sentiment analysis using containerized microservices approach
Word Count:	5080
Page Count:	18

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	31st January 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Improving sentiment analysis using containerized microservices approach

Bharat Goyal
x19215860

Abstract

Disaster management heavily relies on the monitoring of social media data, but the number of users growing is exponential, which generates a very large amount of data when a disaster happens. It is very difficult to monitor the data manually, so to monitor it requires a highly available system that can be scaled up and down easily according to the demand and perform analytics in real-time. This work aims to develop and evaluate a system that solves the issues of collecting such huge data, performs analysis in real-time, and can be set up and scaled very easily. Performance of the system will be measured by comparing memory usage, execution time, the accuracy of the models, and response time of the system. After evaluating the performance of the system, storage requirement was reduced by 31 percent when a parquet file format is used compared to a text file to store, the accuracy of the model was increased by approximately 5 percent and there was almost a 75 percent reduction in response time of the system. However, this system has some limitations which can be addressed and further research can be done to improve the system.

1 Introduction

Natural or man-made disasters, such as cyclones, hurricanes, floods, volcanic eruptions, terrorism, or a pandemic can result in the loss of lives and massive destruction. In the immediate aftermath of a natural hazard event, victims need both immediate and long-term assistance from the community. Many potential solutions are being investigated in support of current disaster management techniques to provide relief to the public, and one such solution is to monitor and analyze social media data. Social media usage in disaster-affected countries results in an enormous amount of unstructured user-generated data, which necessitates the development of new computational techniques. Out of the many computational techniques, sentiment analysis can automatically extract and summarize a vast amount of data that the average human reader is unable to comprehend.

Traditional approaches in developing a sentimental analyzer are by using a monolithic architecture, which researchers often choose. What they do is collect data in text files and then analyze it using standard tools and then share the results, and update them once the next analysis is complete. This is easier to design and implement the project. But if the dataset is huge, it becomes difficult to analyze the data and develop classification models. It is also very difficult to scale up the application quickly as the whole application needs to be deployed on a single server, which becomes costly and time-consuming due to the tight coupling between components. In disaster-affected areas, it becomes highly critical to scale up the application quickly so that disaster management efforts can be quick. Analysis results need to be shared

with a large set of people in real-time such as other government departments, non-government organizations, or public volunteers, so that they can contribute to relief efforts response time of the application needs to be high, which is often not when following this approach. Another approach is to use microservices architecture which enables to break up the functionality into pieces and deploy them separately on different machines. This makes it comparatively easier to scale up the application, but some problems remain, such as implementing load balancing and deploying each service manually. It is also difficult to integrate them with CI/CD pipelines.

Therefore, identifying the correct approach remains an open research area. Serverless computing is one approach that can be considered. No server management is required, reduced costs, high scalability. But the problem with this approach is too much dependency on a single cloud provider, which is also termed as vendor lock-in, difficulty working with cloud APIs, and less control over the underlying infrastructure. Therefore, an approach that uses the strengths of both microservices architecture and serverless architecture, while reducing their limitations, is required. The aim of this is to evaluate if the proposed containerized microservices approach in this work can solve the described issues.

The containerized microservices approach divides each task in the sentiment analysis process into loosely coupled microservices deployed using docker containers so that researchers can easily make changes to a service without affecting another. Each service can also be easily scaled horizontally depending on the requirement using Docker Swarm In this approach, containers can be deployed on a single host or multiple host machines. For instance, Model training, which is a resource-intensive task, can be deployed on host machines with higher specifications, such as a higher number of CPUs or GPUs compared to the data collection service which doesn't require too many resources. This approach is very useful in disaster scenarios as the disasters can be sudden, such as wildfire, storm or an earthquake, so services can be created and scaled up in a very short time to enable public authorities to respond quickly. Another example where this approach will be useful in analyzing social media content is the Covid-19 pandemic. As the pandemic is affecting countries in multiple waves and once the wave starts cases start increasing exponentially, services can be scaled quickly, and when the cases go down, services can be scaled with just a few terminal commands to save on costs.

The docker based microservices approach proposed is used to create a machine learning application that performs sentiment analysis using real-time Twitter data. To evaluate the effectiveness of this, various tests are performed and detailed in the evaluation section.

1.0.1 Research Question

Objective of this research is to answer the following research questions -

1. Does the container-based microservices approach have benefits over other approaches for social media sentiment analysis?

2. Does performing sentiment analysis using real-time data improve the performance of sentiment classifiers compared to just using historical data?

This report is structured as follows. Section 2 presents related works. Section 3 describes the employed methodology. Section 4 discusses the design specification. Section 5 presents the implementation. Section 6 presents the evaluation, and Section 7 conclusions and future work are discussed.

2 Literature Review

The following is a summary of some past writing papers which are identified with the theme. The information gives a concise finish of the work done by different analysts; additionally a short synopsis of the different calculations and methodologies that are being utilized by them.

2.1 Real time twitter sentiment analysis

Yadranjiaghdam et al. (2017) proposes a method for extracting and analyzing structured and unstructured Twitter data using in-memory processing. This research examines the earthquake in Japan and the reactions it elicited from around the world. To analyze data in real-time, the stream of data must be ingested and processed before the data is stored. Real-time results can be achieved with this method, but it cannot accomplish anything more than simple data processing. The suggested framework provides an infrastructure for real-time processing. Machine learning algorithms can be used to analyze real-time data, such as sentiment analysis. A streaming machine learning method is provided by the engine, although it is often necessary to analyze past data. This paper does not evaluate the proposed framework for the accuracy of models but only proposes a method to perform real-time analytics in real-time using Apache Spark, but will be following a different approach. Twitter data will be collected in real-time and stored but instead of using Apache Spark, which is difficult to set up and increases the complexity of the project, python libraries such as pandas, which also does in-memory processing, will be used to implement the project. Historical data will be taken into consideration along with real-time data and then evaluated to evaluate the performance of the architecture. Then the machine learning models for sentiment classification will be evaluated for accuracy and the time taken to execute the model so that a conclusion can be made based on both.

Khaleq and Ra (2018) proposes a microservice architecture for Twitter data analytics during a disaster that meets the requirements of disaster management and a cloud-based microservices Twitter analytics framework for disaster management is proposed. The presentation of a cloud-based microservices architecture framework for Twitter analytics on hurricane disaster data. The framework is composed of Twitter streaming, preprocessing, disaster relevance classification, disaster phase classification, and knowledge extraction. In this paper, a microservices architecture prototype is implemented but doesn't describe how the data is filtered, which machine learning models to implement, or compare the performance of the system to another model. So the research will be extended to include these aspects.

Karanasou et al. (2016) provides high-quality research on real-time Twitter data analytics and presents a scalable system using Apache Storm, which has features such as scalability, parallelism and fault-tolerance. The topology of the system has a node (called Spout) that acts as the source of Twitter data stream. Then there are nodes(called Bolt) that process the data. It has the following types of bolts: the preprocessing bolt, the post-processing bolt, the classification, bolt, the NoSQL bolt, and the statistics bolt. Architecture is very complex to implement real-time analytics and system requirements are high. The research will be presented in this paper to implement a real-time Twitter sentiment analysis that is not as complex to set up and can run with low system requirements.

2.2 Machine Learning Methods

Aziz et al. (2019)analyzed ten different natural disasters through tweets using sentiment analysis and deducted the most dominant disaster. They used sentiment analysis techniques to

classify Twitter data related to natural disasters. In this paper, an approach was described to classify tweets as positive, neutral, or negative using the Naive Bayes algorithm. Data is extracted from Twitter using the Twitter API search feature, but we will be using Twitter streaming to collect the data. The solution presented in the paper is implemented in R language, but we will use Python programming language. Also, this paper doesn't compare the presented model with any other model and mostly focuses on visualization, so we will be evaluating another machine learning algorithm and metrics will be compared. This paper also doesn't present an approach to scaling up the system, but we will be evaluating ways to scale up the system using the microservices approach.

Maldonado et al. (2016) presents a system for monitoring Twitter, connecting to the API, and filtering content based on four categories (volcanic, telluric, fires, and climatological) that affect Ecuador due to its geographic location, and all tweets are stored in a database for analysis. The NLTK tool is used to determine the frequency of a word in a tweet, which is then used to classify it into one of the recommended categories. The dashboard is implemented with PHP which adds an overhead of using another programming language with different requirements for using the libraries but will be using a single programming language throughout the system which doesn't add this overhead. There is also no performance evaluation of the dashboard presented. Also, the paper presents a way to filter, clean, and process the data from a database but intermediate data is not stored anywhere, and the whole system is dependent on the previous steps; but our work will follow the microservices approach and each step of the process will not be dependent on any other step.

2.3 Microservices architecture

Malik et al. (2020) proposes a good system for containerization of airline sentimental analysis using docker containers. It divides the process into multiple steps using separate containers and presents a frontend system developed using Java. Similarly to the above paper, this adds an overhead of using an additional language. It also doesn't provide in detail the implementation of the system, which makes it difficult for researchers to implement without doing some extra research. It also doesn't show how the data can be collected in real, stored and processed and also how the system will be scaled up when the number of users grows. Why the approach was better was not presented in this paper with proper evaluation results. Our research will be aimed to show how this architecture can be improved by using different approaches of collecting the data, reading the data, cleaning, processing the data and finally generating the results all in real-time.

Liu et al. (2020) presents an overview of microservices and compares them with traditional monolithic and service-oriented architecture. It explains the technology of the container system and virtualization concepts are used in the container system in good detail. It then presents how the microservices can be containerized and how containers can be orchestrated using different tools. It gives an overview of containerization and how it can be used in creating microservices. It finally presents the challenges of containerization,n such as how networking factors affect the performance of the system. Debugging challenges in the distributed environment and how they can implement is also presented in this paper. This research was used to make decisions about the different tools and architecture of the system presented in this paper.

Hamilton et al. (2020) presents an intelligent system for big data applications and presents an Apache Spark-based micro-service orchestration framework that extends database operations. Architecture takes full advantage of the cluster, thread, and, asynchronous parallelism. Finally, they have presented a containerized system to reduce network overheads. Spark cluster

is used to directly interact with source database store and then outputs the data in a sink data store. The approach in our research is different in that the output of the analytics system will be stored in the storage so instead of having to maintain multiple databases. Also, the data pipeline has been created in the presented research instead of having to deal with an advanced system such as apache spark.

2.4 Summary

Reference	Paper title	overview	Difference with our approach
Yadranjiaghdam, B., Yasrobi, S. and Tabrizi, N. (2017).	Developing a real-time data analytics framework for twitter streaming data	Proposes a method for extracting and analyzing structured and unstructured Twitter data using in-memory processing, stream of data must be ingested and processed before the data is stored	Twitter data will be collected in real time and stored but instead of using apache spark, python libraries such as pandas dataframe which also does in memory processing will be used to implement the project. Historical data will be taken into consideration along with the real time data and then the performance of architecture will be evaluated.
Khaleq, A. A. and Ra, I. (2018).	Cloud-based disaster management as a service: A microservice approach for hurricane twitter data analysis	Proposes a microservice architecture that meets the requirements of disaster management and a cloud-based microservicestwitter analytics framework is proposed	A microservices architecture prototype is implemented, but doesn't describe how the data is filtered, which machine learning models to implement, and compare performance of the system to another model, so these aspects will be included.
Karanasou, M., Ampla, A., Doulkeridis, C. and Halkidi, M. (2016).	Scalable and Real-Time Sentiment Analysis of Twitter Data	A scalable system using Apache Storm, Topology of the system has a node (called Spout) that acts as the source of Twitter data stream. hen there are nodes(called Bolt) which processes the data	Output of analytics system will be stored in the storage instead of having to maintain multiple databases. Also the data pipeline has been created instead of having to deal with advanced system such as apache spark.

Figure 1: Real time analytics

Reference	Paper title	overview	Difference with our approach
Aziz, K., Zaidouni, D. and Bellafkih, M. (2019).	Social network analytics: Natural disaster analysis through twitter	Approach was described toclassify tweets as positive, neutral, or negative using Naive Bayes algorithm	Implemented in R language but we will use python. Paper doesn't compare the presented model with anyother model and mostly focuses on visualization, so we will be evaluating another machine learning algorithm and metrics will be compared. This paper also doesn't present any approachto scale up the system but we will be evaluating ways to scale up the system.
Maldonado, M., Alulem	System for monitoring natural disasters using natural language processing in the social network twitter.	System for monitoring twitter, connecting to the API,and filtering content, all tweets are stored in a database for analysis. The NLTK tool is used to determine the frequency of a word in a tweet, which is then used to classify it into one of the recommended categories.	Dashboard is implemented with php but we will use a single programming language throughout the system There is also no performance evaluation of the dashboard. Intermediate data is not stored anywhere and whole system is dependent on the previous steps but our work will follow microservices approach using containers.

Figure 2: Machine Learning methods

Reference	Paper title	overview	Difference with our approach
Malik, S., El-Sayed, H., Khan, M. A. and Alexander, H. (2020).	Application of containerizedmicroservic e approach to airline sentiment analysis	Proposes a good system for containerization of airline sentimental analysis using docker containers. It divides the process into multiple steps using separate containers and presents a frontend system developed using Java.	It also doesn't provide in detail the implementation of the system. Doesn't show how the data can be collected in real, stored and processed and also how the system will be scaled up when the number of user grows. Our research will be aimed to show how this architecture can be improved by using different approaches of collecting, reading, cleaning, processing the data and finally generating the results all in real time.
Liu, G., Huang, B., Liang, Z., Qin, M., Zhou, H. and Li, Z. (2020)	Microservices: architecture,container, and challenges	presents an overview of microservices and compare them with traditional monolithic and service oriented architecture. It explains the technology of the container system and virtualization concepts are used in the container system in good detail.	This research was used to make decisions about the different tools and architecture of the system presented in this paper.
Hamilton, M., Gonsalves, N., Lee, C., Raman, A., Walsh, B., Prasad, S., Banda, D., Zhang,L., Zhang, L. and Freeman, W. T. (2020).	Large-scale intelligent microservices	Presents an Apache Spark-based micro-service orchestration framework that extends database operations. Architecture takes full advantage of cluster, thread, and, asynchronous parallelism.	Approach in our research is that the output of analytics system will be stored in the storage. Also the data pipeline has been created in our research instead of having to deal with advanced system such as apache spark.

Figure 3: Microservices Approach

3 Research Methodology

To conduct the research many steps were followed as Data Analysis and machine learning is a multi-step process. Each step in the research process has been described in the detailed below.

3.1 Data Collection

Twitter data is collected using tweepy python library to access Twitter API. Twitter data can be collected by either using Twitter API search or streaming. There are various limitations of using the search such as fetching 450/15 minutes so using the search feature is not very useful if a large dataset is required for the project. Twitter streaming provides access to real-time tweets and has no such limitation. For this project, a python script has been written to stream real-time data and store it in SQLite database. To access the Twitter API using tweepy, credentials need to be created from Twitter developer portal access which can be requested from Twitter. Once the access is granted an app can be created on the developer portal and credentials can be generated to set up the Twitter data stream. To get the tweets related to disasters list of keywords has been used to filter data.

3.2 Data Storage and Reading

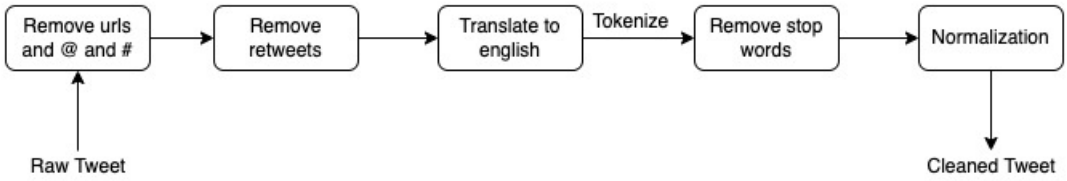
Data will be read from SQL database using a python script into a DataFrame. As the Data-Frames are in memory it needs to read the data multiple times when performing analysis and machine learning operations. It may not affect the project when the data size is small but when there are millions of records to read it can make a huge difference in overall analysis

time. Another alternative to Pandas DataFrames are Dask DataFrames, which is a large parallel DataFrame, composed of many smaller Pandas DataFrames, split along the index.

For Twitter data storage SQLite database will be used, which is a fast, end program database storage technology that stores the data on the disk and can use. Using cloud-hosted databases such as Amazon Relational Database Service (RDS) is a much better approach in a microservice architecture. It is also decoupled completely from the architecture because there is no dependency on the script because of high availability and scaling abilities but due to the time limitations of this research, data will be stored using SQLite, because it is much easier to set up and integrate code with. Some file systems are also used to collect data such as CSV. As a part of this research read times of SQL, CSV, Pickle, and Parquet by both Pandas DataFrame and Dask DataFrame will be evaluated so that researchers can make an informed decision about which technology to use according to their needs.

3.3 Data Cleaning and Processing

As the data collected in dataframe is raw it needs to be cleaned before it can be processed. First, the URLs are removed and then @ and hashtag are removed as they do not help get the sentiment of the text. Then the RT symbol is removed because it stands for retweets in the twitter text. Data collected also contains tweets of other languages which needs to be converted to a standard language so they are converted to English using a textblob library. At the last step stop words, words like 'if', 'but', 'we', 'he', 'she', and 'they', are removed from the tweet after converting the words into tokens and then rejoining them back. Texts are then normalized which means converting different forms of a word into a normal form using the stemming approach.



Once the data is cleaned it is assigned a target variable against which algorithm prediction can be trained and tested which is done at this step. It is assigned a variable 1 for positive, variable 0 for neutral and -1 for negative. But any variable can be assigned.

3.4 Algorithm Training and Prediction

For this research, two classification algorithms are trained and predictions are performed. First is the Naive Bayes algorithm in which a particular feature of a class is assumed to be unrelated to the presence or absence of any other feature in the class. Next is logistic regression which predicts the probability of a target variable. Data is then split into test data and train datasets. The training dataset is used to train the classifier and the prediction is performed on the test dataset. The accuracy of the models can then be checked for the algorithms. In this research as the data is being collected in real-time, the accuracy of the algorithm is calculated so that classifier is improved in real-time. In the evaluation sections, accuracy is being compared for both the algorithms at an increasing number of records to evaluate if performing the analytics in real-time results in accuracy improvement of the classifier.

3.5 Visualization

For visualization of the results a live dashboard has been created. Flask framework has been used to setup the dashboard as it provides an easy way to setup endpoints and templates can be used to generate web pages pages if a very simple dashboard needs to be setup.

3.6 Containerized microservices

The whole process is split into four scripts for this project and then each script is containerized using packaged into a container using docker and then deployed on a docker swarm cluster. Another approach to scaling up the microservice architecture is by using a container-based approach and another by using different servers for different services.

As a part of this research, dashboard performance is tested before and after scaling up the system to see if there is any performance improvement.

4 Design Specification

This section details a sentimental analyzer using containerized microservices approach and architecture diagram is presented in figure.

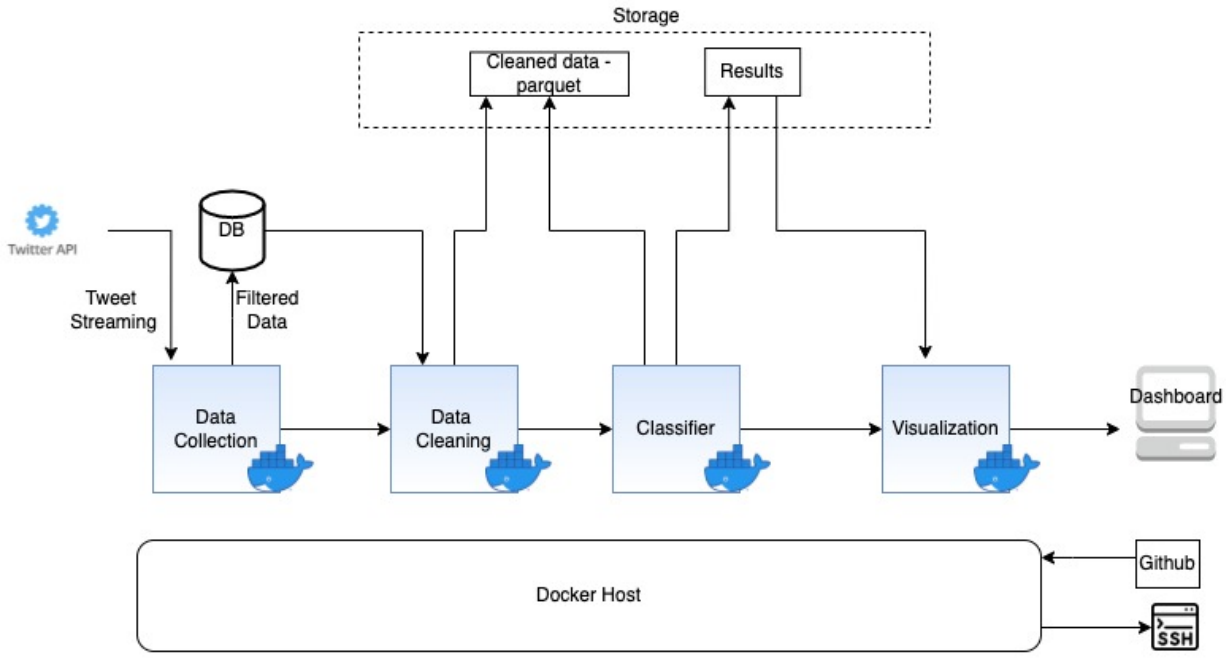


Figure 4: Proposed archiecture

4.1 Microservices architecture

Docker engine is used to create the containerized microservices, which divides each task in the sentiment analysis process into loosely coupled microservices so that the researchers can easily make changes to a service without affecting another. Each service can also be easily

scaled horizontally depending on the requirement. Other containers such as windows containers can be used depending on the workload and processing requirements but this research will be limited to using docker containers. Containers can be deployed on a single host or multiple host machines. For instance, Model training which is a resource-intensive task can be deployed on host machines with higher specifications such as a higher number of CPUs or GPUs compared to the data collection service which doesn't need require too many resources. This approach is also very useful in disaster scenarios as the disasters can be sudden such as wildfire, storm or an earthquake, so services can be created and scaled up in a very short time to enable public authorities to respond quickly. The covid-19 pandemic is affecting countries in multiple waves and once the wave starts cases start increasing exponentially, so this approach can also be implemented to monitor social media and provide relief efforts quickly, and when the cases go down services can be scaled down quickly to save on costs.

4.1.1 Data Collection

Data Collection service streams the live Twitter data using tweepy according to the filter criteria, disaster-related keywords are used as a filter and then stores the data in a database.

4.1.2 Data Cleaning

Data Cleaning service cleans the data collected as the Twitter data can have a lot of unnecessary information such as removing links URLs, punctuations, etc and stores the data as a parquet file on the storage server which can then be used for transformation and model training.

4.1.3 Classifier

Model training service transforms the data into a vector and then classification models are trained. Prediction is performed and then tested for accuracy for which the results are stored on storage.

4.1.4 Visualization

Lastly, the visualization service creates a live dashboard using the results on storage which can then be accessed remotely.

4.2 Storage

For storage, centralized storage is used which can be mounted on the docker containers using bind mounts. As the docker containers are ephemeral they are not very useful to store data and it needs to be stored in a storage server. Storage is decoupled from the containers because if we store the data on containers then it will be lost once the docker containers are deleted.

4.3 Scaling microservices

Once the services are required to be scaled up a container orchestration service is required. In the proposed approach Docker Swarm, which is a container orchestration tool is as opposed to Kubernetes. Docker Swarm is a good choice if the cluster size is small as it comes built-in the docker engine and is easy to set up and deploy compared to Kubernetes. To create a docker swarm cluster master workers are created and the services are deployed on the worker nodes.

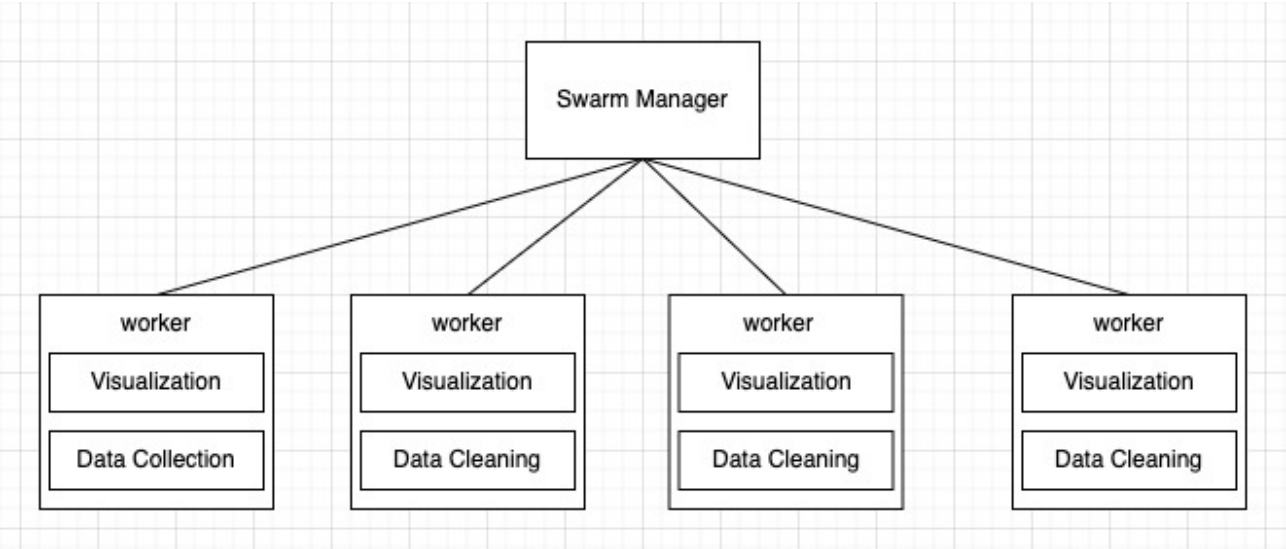


Figure 5: Docker Swarm Cluster

5 Implementation

Several python scripts are developed for this project which are run inside docker containers once the stack is deployed on the machine.

5.1 Python scripts

5.1.1 Data Collection

Data Collection script streams the live twitter data using tweepy according to the filter criteria, disaster-related keywords such as ”storm”, ”hurricane”, ”wildfire” are used as a filter then stores the data in a database. Only Tweet ID and tweet text are stored to speed up the collection process but other data related to tweets can also be stored. A python script has been developed to stream real-time data and store it in SQLite data which starts collecting data automatically once the container is up and running. To access the Twitter API using tweepy, credentials need to be created from Twitter developer portal access which can be requested from Twitter. Once the access is granted an app can be created on the developer portal and credentials can be generated to set up the Twitter data stream.

5.1.2 Data Cleaning

Data Cleaning python script reads the data from the database using DataFrame and cleans the data collected as the Twitter data can have a lot of information not useful for analysis and training the model. It removes the null values from the data first, then URLs, punctuation, @ references and hashtag symbols are removed, and then stop words are removed from the data. Once the data is cleaned it is stored as a parquet file in the storage which can be further used for transformation and model training.

5.1.3 Classifier

In this python script, the Parquet file is read using DataFrame. A feature vector is then generated using the TfidfVectorizer library imported from sklearn and the data is transformed in the next lines. X is the NumPy array of feature vectors the Y is the NumPy array of target variables which we will try to predict. Two classification models Naive Bayes and Logistic Regression are trained in this script. Pickle is used to serialize both classifiers and then the serialized format is saved to a pickle file. Dataset is split into 80:20 for training and testing respectively, and Prediction is performed using the test data. Accuracy is determined for both classification models whose results are stored on the storage to be used for creating a live dashboard.

5.1.4 Visualization

The dashboard is created using flask API which makes it easy to develop API with one endpoint. Templates are used instead of a frontend library to speed up the creation of a dashboard for the prototype. Matplotlib library has been used to generate plots.

5.2 Docker files

Dockerfile has been used to create the container for each service. In the dockerfile python:3.9.7 image has been imported which comes with preinstalled python on Linux container. WORKDIR /app is created which will be used as the main directory inside the container. Requirements file which is generated by using pip freeze command is then copied and installed. In the last line python script is run for the respective container. Dockerfiles are named according to the script so that it is easier to recognize them.

docker-compose file is used to create a microservice without having to run each container manually. It is in YAML format. Storage is mounted to the container using a bind mount.

5.3 Tools Used

Python v3.9.0 was used to develop the whole project. Data analytics and machine learning libraries are Numpy, Pandas, NLTK, scikit-learn, Flask python library was used to develop API. Docker was used to containerize the project.

5.3.1 Apache Bench

To test the scalibility of the application Apache Bench (ab) is used. It very simple is a load testing and benchmarking tool for HTTP servers. It can be run from the command line. One minute is all it takes to get a quick load testing result. Because it doesn't require a lot of knowledge of load and performance testing concepts, beginners and intermediate users can use it. No complicated setup is required to use this tool.

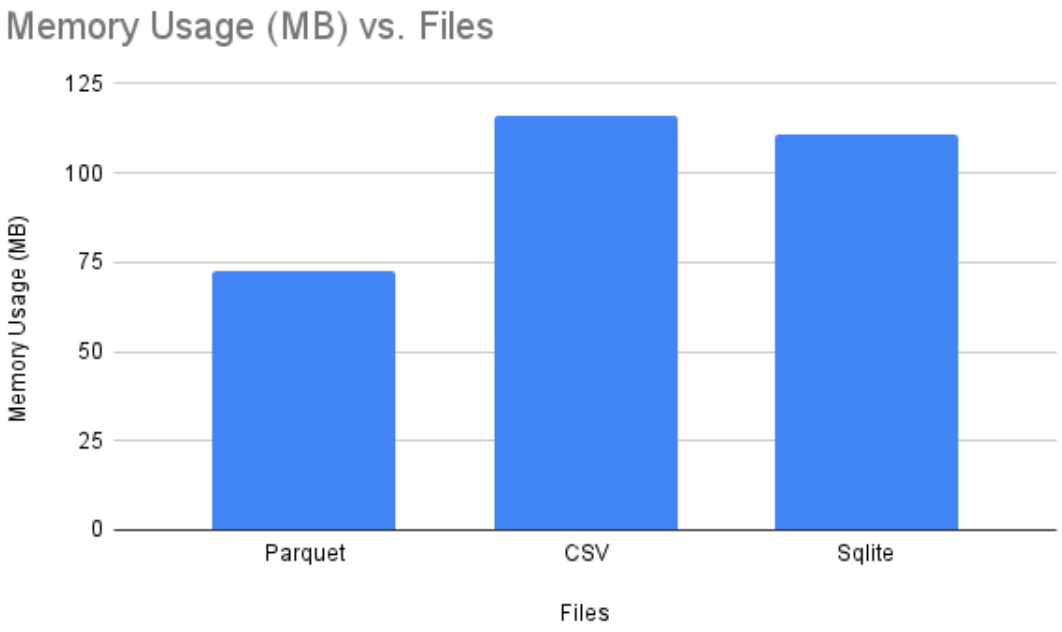
6 Evaluation Results

research evaluation section is divided into three parts.

6.1 Data Read evaluation

6.1.1 Evaluation of file sizes

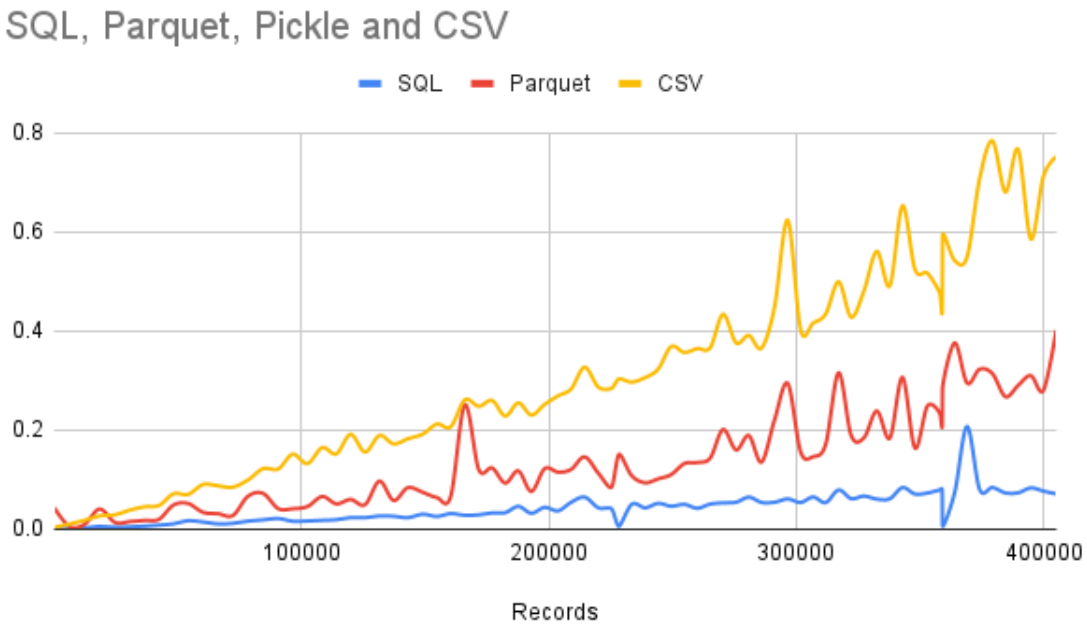
For this evaluation size of the SQLite, CSV, and Parquet on files are compared for the same number of records.



Parquet files need less storage compared to the CSV file, so storage costs can be reduced especially for large datasets.

6.1.2 Evaluation of Pandas Dataframe Read times

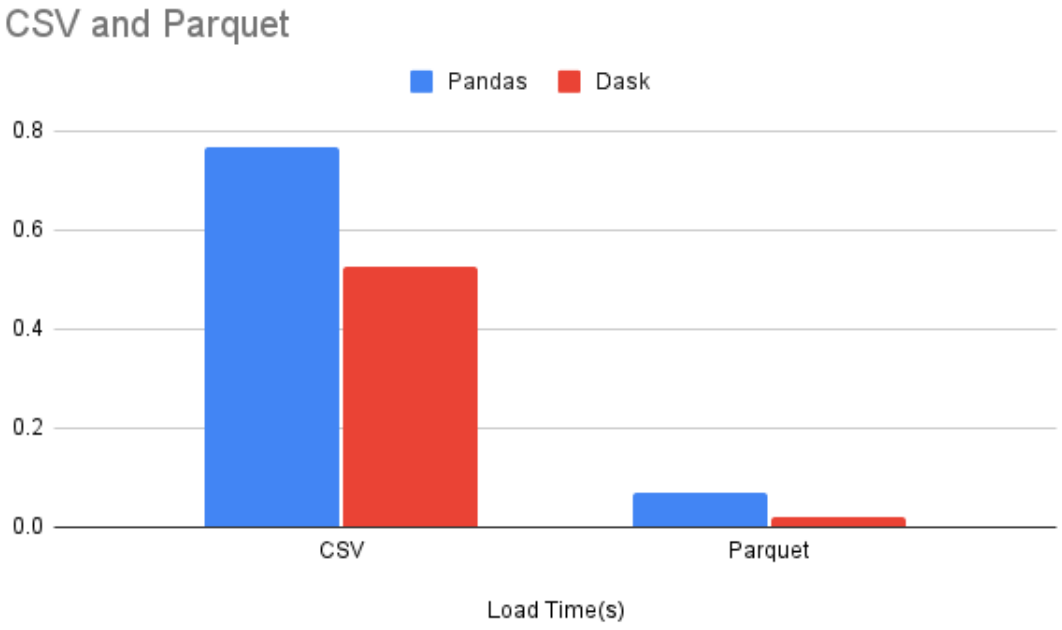
For this evaluation, SQLite, Parquet and CSV are read by Pandas Dataframe for different numbers of data records. An evaluation script was run while collecting the data to record the read times and a line chart is plotted as shown in the figure.



It can be seen from the graph that Dataframe read time for SQLite database is significantly lower than Pickle, Parquet, and CSV especially when the number of records is more but it is costly to host and manage a production database so file storage may be a better option for researchers for performing Twitter sentiment analysis. In file storage comparison Parquet performs better in comparison with CSV supported so for very large datasets parquet file is a much better option as it stores data in a columnar format instead of CSV which stores the data in table structure, so read times can be improved further when we only need to read some of the columns.

6.1.3 Comparison between Read times - Dask Dataframe and Pandas Framework

In this evaluation CSV and Parquet files containing the same number of records are loaded to both Dask and Pandas dataframes and then the read times(seconds) for both files are compared.

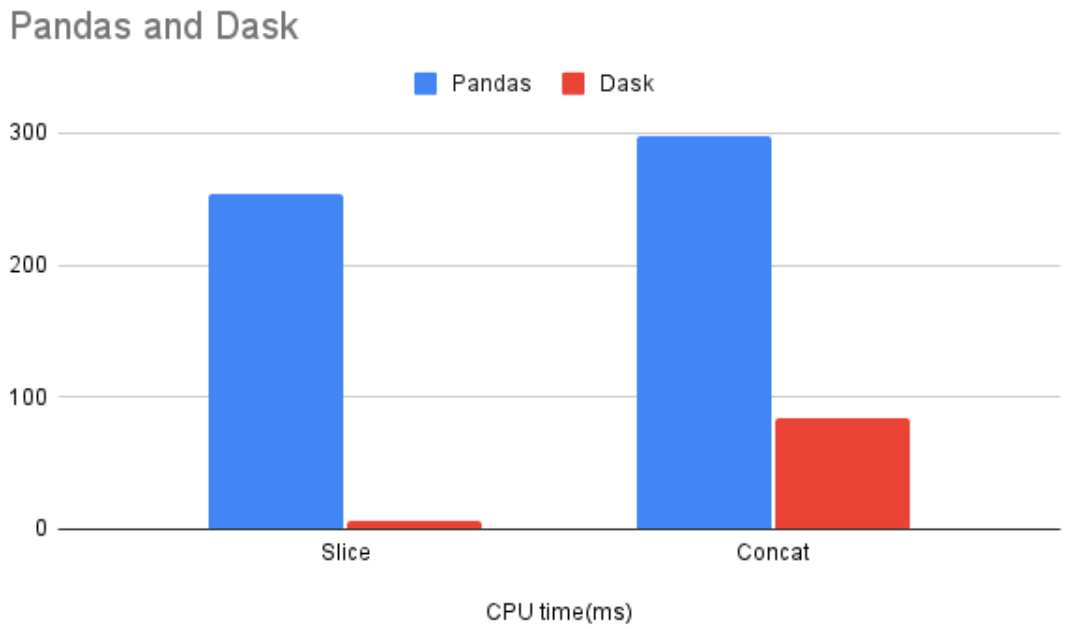


It can be noted that Dask can load both CSV and Parquet files much faster than the Pandas Dataframe

To conclude the first part of the evaluation SQL is preferred due to the faster read time but when it's not possible to use a database, Parquet file is better in terms of both file size and read times compared to a CSV file and dask dataframe are better in terms of reading the reading a very large dataset as it breaks it chunks of pandas dataframes but

6.2 Dataframe operations evaluation

For these evaluations, a large dataset with 1.6 million records is taken to evaluate if using Dask Dataframe make a difference in execution time of algorithms and accuracy. Code is run ten times in parallel using docker in each scenario and average values are taken so that error in values can be reduced.



6.3 Algorithm comparison

For this evaluation, both naive bayes and logistic regression classifiers are trained and tested to find for an increasing number of records. We can see from both the graphs that as the number of records starts to increase the accuracy of the model starts to increase. When we compare the accuracy of both models, Logistic regression is higher compared to Naive Bayes.

Figure 6: Accuracy comparison

6.4 Performance benchmark of microservices

For this evaluation, Apache Bench is used to benchmark the microservices. First tests are conducted with 1 container for the dashboard and then the same tests are conducted are after scaling up the service to 10 containers. 10000 requests are sent to the server with 0, 20 and 100 concurrent requests(number of users) and the average time is noted for each request is noted. The results are shown in figure

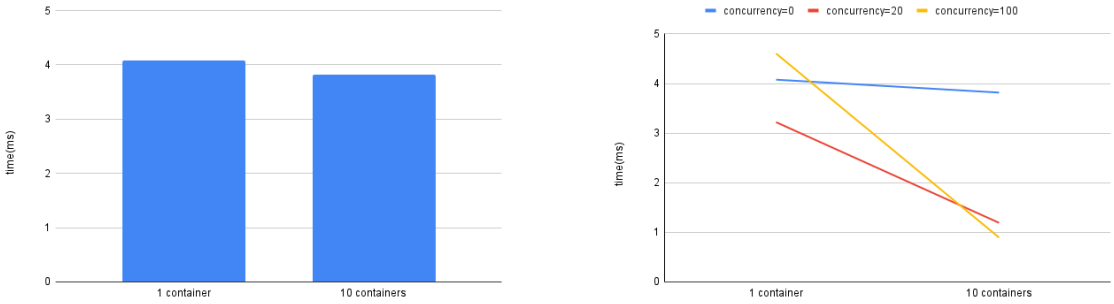


Figure 7: Apache bench load test results

As we can see from evaluations results using multiple containers reduce the response time when no concurrent requests are sent but there is a drastic reduction in response time when the concurrent requests are sent. As we increase the number of concurrent requests the difference starts increasing. For 20 concurrent requests there is a 63 percent reduction in response time but for 100 concurrent requests reduction is 80 percent which is

7 Conclusion and Future Work

In this work, we discussed the container-based microservices approach of developing a sentimental classifier for a disaster scenario. To achieve these recent developments microservices architecture were researched and a real-time sentimental classifier was implemented with a working live dashboard using containers. As evaluated the system shows the reduction in storage requirement, improvement inaccuracy of the model and average response time. But there are some limitations such as complexity in understanding virtualization technology, the ephemeral nature of containers which means everything can not be containerized, and difficulty in monitoring the containers and securing the system which was not evaluated in this research. In future work, these aspects can be researched and implemented in the architecture.

References

Aziz, K., Zaidouni, D. and Bellafkih, M. (2019). Social network analytics: Natural disaster analysis through twitter, *2019 Third International Conference on Intelligent Computing in Data Sciences (ICDS)*, pp. 1–7.

Hamilton, M., Gonsalves, N., Lee, C., Raman, A., Walsh, B., Prasad, S., Banda, D., Zhang, L., Zhang, L. and Freeman, W. T. (2020). Large-scale intelligent microservices, *2020 IEEE International Conference on Big Data (Big Data)*, pp. 298–309.

Karanasou, M., Ampla, A., Doulkeridis, C. and Halkidi, M. (2016). Scalable and real-time sentiment analysis of twitter data, *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pp. 944–951.

Khaleq, A. A. and Ra, I. (2018). Cloud-based disaster management as a service: A microservice approach for hurricane twitter data analysis, *2018 IEEE Global Humanitarian Technology Conference (GHTC)*, pp. 1–8.

Liu, G., Huang, B., Liang, Z., Qin, M., Zhou, H. and Li, Z. (2020). Microservices: architecture, container, and challenges, *2020 IEEE 20th International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 629–635.

Maldonado, M., Alulema, D., Morocho, D. and Proaño, M. (2016). System for monitoring natural disasters using natural language processing in the social network twitter, *2016 IEEE International Carnahan Conference on Security Technology (ICCST)*, pp. 1–6.

Malik, S., El-Sayed, H., Khan, M. A. and Alexander, H. (2020). Application of containerized microservice approach to airline sentiment analysis, *2020 14th International Conference on Innovations in Information Technology (IIT)*, pp. 215–220.

Yadranjiaghdam, B., Yasrobi, S. and Tabrizi, N. (2017). Developing a real-time data analytics framework for twitter streaming data, *2017 IEEE International Congress on Big Data (BigData Congress)*, pp. 329–336.