Smart City Service Monitoring Using Twitter Analytics

Venkata Krishna Kota
Central Research Laboratory
Bharat Electronics Limited
Bangalore, India
venkatakrishnav@bel.co.in

Venkateswarlu Naik B
Central Research Laboratory
Bharat Electronics Limited
Bangalore, India
venkateswarlunaikb@bel.co.in

Vasudeva Rao Prasadula
Central Research Laboratory
Bharat Electronics Limited
Bangalore, India
vasudevaraoprasadula@bel.co.in

DOI: 10.29322/IJSRP.9.08.2019.p92136
http://dx.doi.org/10.29322/IJSRP.9.08.2019.p92136

Abstract—Smart City domain has multiple verticals like Water Management, Garbage Management, Power Management, Transportation Management etc. Such services will have to find problems in their respective domain and react to it in less time. An application is proposed in this paper to collect the relevant information from social network data, to process and filter the received relevant information and to communicate about it to the relevant department to handle the issues. The proposed application is scalable and fault tolerant.

Keywords- Smart City; Twitter Analytics

INTRODUCTION

Smart city has multiple services. There is a need for an automated system to monitor the service failures and to report them to respective personnel. One such source for this task is from user complaints. Always users will not complaint to respective departments. It may be of many reasons like, lack of information about where to, whom to and when to complaint or the complaining mechanism may not be user friendly, or it may be of any other reason. Thus people may not complain about it directly to the respective departments always. But it is so common that people will talk about it with friends, and post about it in their favorite social network.

So the information regarding service failure may not be available directly to the respective departments. Departments need to extract relevant information from other sources where such information exists.

Social networks are important such information sources. People more frequently post information regarding smart city service failures in social networks. Most famous such social networks are Twitter, Facebook, and WhatsApp .etc. Extracting and processing data from such social networks will help to monitor and report the smart city service failures.

This paper proposes a method and application to read data from twitter and extract information relevant to smart city service failures and to report them to respective departments of smart city. So each department will receive the information (which is extracted from twitter data) regarding their service failures on their dashboard and they may take necessary action.

Section 2 describes some of the related work done in this area. Section 3 describes the technology selection details to realize the proposed work. Section 4 describes approach of the proposed work. Section 5 describes the system architecture and design methodology of the proposed system. Section 6 provides the implementation details of the proposed system.

RELATED WORK

This section describes some of the related work done by the research community.

Angel Martin, et al. proposed a method to analyze the tweets to depict events in the city and to discover their spatiotemporal characteristics [1].

Sugimiyanto Suma, et al. proposed a model to analyze twitter data to detect and validate spatiotemporal events in London. With their method they could find congestion around the city, they could also detect the occurrence of cultural events happening inside the city without having prior information [2].

Arthur Souza, et al. proposed a case study for real time capturing of crime event information from social media messages. They have also proposed a platform for fast processing and visualization of crime data from Twitter tweets [3].

Li Mengdi, et al. proposed a framework for sentiment analysis. It collects, preprocesses, analyses and maps sentiment of people from Twitter. It helps the Government to monitor their citizens' moods [4].

Many researchers have analyzed the twitter messages and brought insights in different applications. In this paper, twitter messages are analyzed for monitoring the services of smart city.

TECHNOLOGY SELECTION

The scope of this paper is to design and implement an application to monitor smart city services from social network data and to report the relevant information to the appropriate departments. In order to do it we need to choose appropriate data source and technologies to implement it.

Choosing Social Network Data Source:

For any data analytics application choosing the appropriate data source will be the key differentiator to its success. V's of big data have to be examined while choosing the data source.

Volume: It should be able to provide huge volumes of data

Velocity: It should be able to generate data with higher rate

Variety: It should be able to produce heterogeneous data types: text, image, audio, video etc.

One more major requirement to select the data source is that that data source should be freely available to the developers. It means it should be open source.

Another major requirement is that there should be a handy API or Connectors to capture the data from that data source.

By considering several social network data bases and by examining their suitability to the above requirements, we have chosen "Twitter" as our data source [7]. Twitter is a popular social network. It generates huge volumes of different varieties of data with high velocities. It provides open source APIs and connectors to capture its data. It also provides APIs to capture the real time streaming of tweets. It also provides pluggable connectors to fit into big data frameworks. Twitter analytics has rich developers as an added advantage. Moreover it has been widely used by many researchers to solve other interesting problems. These features of twitter encouraged us to select it as our data source.

Choosing Big Data Processing Framework:

In order to address the 5 V's (volume, Velocity, Variety, Value and Veracity) of big data, it is decided to use big data processing framework to implement the core business logic. Current requirement needs stream processing capability, complex event processing capacity and a handy interface to consume tweets. Big data processing frameworks like hadoop map-reduce, apache spark and apache flink are explored. Apache Flink got little edge over the other two because of its following features.

- Distributed Processing Capability
- Stream Processing Capability and Strong Data Stream API
- Complex Event Processing Capability
- Open Source Nature
- Fault Tolerance and High Scalability
- Low Latency and High Performance
- Handy Connector to consume data from Twitter
- Handy Connectors to read / write data to messaging applications like 'Kafka'

Because of the above features, Apache Flink [5] is chosen as the data processing framework for current application.

Choosing the Message Broker:

Message brokers play an important role in big data frameworks. In the current application, we need to store huge number of tweets coming with high velocities. So the message queue should be capable of storing the messages in distributed way across all the nodes of the cluster. As already specified twitter produces variety of data (text, image, audio and video). So the message queue should be capable to store messages with heterogeneous schemas. The messaging system should also have the capability to logically segregate the messages into different topics. It should also facilitate efficient publish-subscribe mechanism to write / read messages to relevant topics. It should read /write message with low latency. The messaging framework should work hand in hand with the data processing framework. Some of the message brokers are examined and Apache Kafka [6] is chosen for its following unique features.

- Better Throughput
- Built-in Partitioning
- Replication
- Fault-Tolerance

Choosing Web Technologies:

Following web technologies are selected for the current application

- Web Server: Apache
- Application Server: Tomcat
- Web UI: HTML5, JavaScript
- Communication protocol: Web Sockets

Current application mainly focuses on real time analysis of tweets and alerting appropriate personnel with relevant information through its web UI. It does not store the messages in the persistence data base. For that one can use one of the NoSQL databases.

APPROACH

Identify the various smart city verticals whose services have to be monitored. Create a Kafka topic for each of them. Prepare a set of relevant keywords for each of those verticals.

To get access to the tweets, one has to enroll for twitter developers [8]. Once it is done, credentials will be provided to access twitter data. Flink has a twitter source which is a handy connector to consume tweets. Using that connector get the twitter streams. Then Flink will be able to receive tweet streams. For each tweet in the stream, extract the location and text information. If the location is within the smart city region, then examine the text of the tweet. If the text of the tweet contains any keywords which are of relevance to any of the smart city verticals, then select that tweet and treat it as relevant tweet. Twitter has a Kafka sink which facilitates to write flink data streams to Kafka topics. Those relevant tweets will be written into appropriate Kafka topics. Thus the flink jobs acts like publishers to Kafka topics.

On the other side, java application running inside server will act like consumer to Kafka topics. It continuously monitors the Kafka topics and consumes messages from them. The consumed messages will be sent to the appropriate WebUI which is tied up with that corresponding Kafka topic. It will send these messages over web sockets. From the client side, each smart city department will have a web UI developed in HTML5 and JavaScript. It will display the messages received through web socket.

Thus the flink job first filters the relevant tweets by matching location of the tweet and tweet text. Filtered relevant tweets will be written into relevant Kafka topics based on the keywords in the text of the tweet. Java application running inside web server continuously consumes those Kafka messages in respective Kafka topics and streams them to appropriate departments using web sockets. Thus each department gets the tweets of its relevance. The user interface will display the messages it has received through web socket on its web page.

SYSTEM ARCHITECTURE

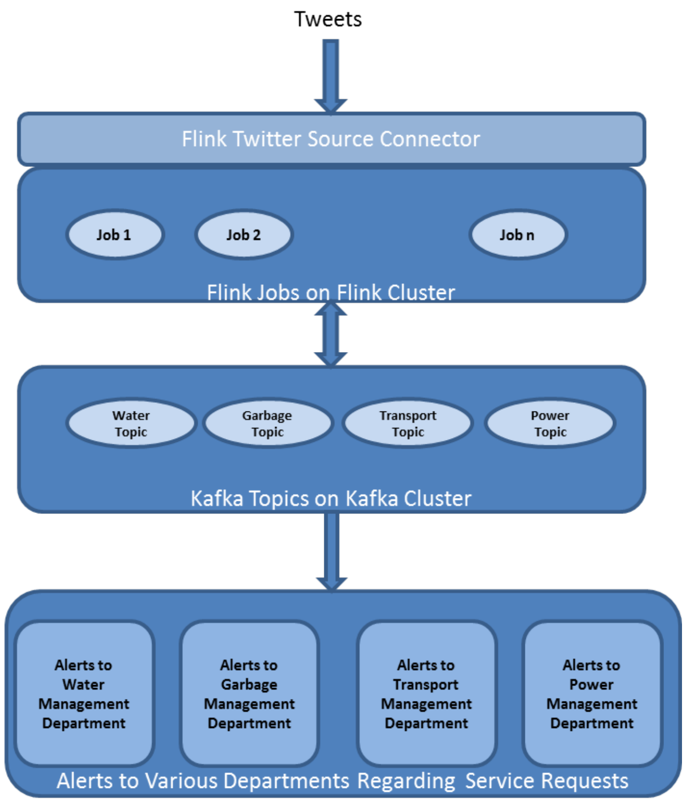
Tweets
Flink Twitter Source Connector
Job 1
Job 2
Job n
Flink Jobs on Flink Cluster
Water Topic
Garbage Topic
Transport Topic
Power Topic
Kafka Topics on Kafka Cluster
Alerts to Water Management Department
Alerts to Garbage Management Department
Alerts to Transport Management Department
Alerts to Power Management Department
Alerts to Various Departments Regarding Service Requests

Figure 1: System Architecture

In this section, architecture is proposed to monitor smart city service failures and to alert appropriate departments about the failure. The system architecture will be as shown in Fig 1.

First developer has to register with “Twitter Developer Account” to get access to tweets. Flink has a Twitter Connector. It helps flink to read data streams from twitter source. Top module in figure 1 describes the capturing of twitter data using Flink’s Twitter Connector.

Different verticals of smart city are depicted as different kafka topics. For example, Water Management, Traffic Management .etc. Each Kafka topic is intended to store messages relevant to that department. Kafka topics on kafka cluster is described by the middle module in figure 1.

Every department of smart city services will have its own dashboard to visualize the service failure related information. Flink jobs will first cosume the twitter data from twitter connector, analyze them, filter them based on the location of the tweet and keyword exists in the tweet. Then it seggregates the tweets based on their relevance to different departments of smart city and writes them to their corresponding kafka topics. Web technologies will read from those kafka topics and display them on the respective department’s dash board. The dashboards are depicted in bottom module.

Actual implementation details and results are described in the following section.

IMPLEMENTATION DETAILS

This section focuses on the implementation details of the application.

Registering for Twitter Developer Account:

Twitter provides controlled access to its APIs and tools. To access its APIs and tools, developers have to register for Twitter Developer Account at “https://developer.twitter.com”. Once it is done, twitter will provide following access credentials to the developer.

- CONSUMER_KEY
- CONSUMER_SECRET
- TOKEN
- TOKEN_SECRET

Developer has to specify those access credentials in their program. If the credentials are valid then twitter will allow that program to read tweets from internet.

Kafka Cluster Setup:

Apache Kafka is a distributed stream platform. It is used for building real time data pipelines and streaming apps. Kakfa can scale horizontally on multiple nodes. In this experiment, mutiple virtual machines with Ubuntu 16.04 are taken and Zookeeper and Kafka is installed in them. Kakfa cluser setup is described in [kafka cluster setup]. Multinode Kafka Cluster is setup by configuring zookeeper and kafka properties as described in [kafka cluster setup]. Once the Kafka Cluster setup is done, sample topics are created in that. The cluster is tested with console producers and console consumers. It has produced messages into kafka topics and consumed those messages as expected.

Creation of Kafka Topics and Relevant set of Keywords for Smart City Services:

Once the Kafka Cluster is setup, next job is to create appropriate topics which are of relevance to the smart city services. Topics for Water Management, Traffic Management, Power (Electricity) Management and Garbage Management are created in the kafka. Each of these topics will contain messages relevant to its domain.

Every domain will have some keywords and phrases which are of relevant to that domain. For example, for Traffic domian, “traffic jam”, “accident”, “road block” .etc are relevant. If text contains some of these keywords, it may be relevant to that domain. Relevant set of keywords are prepared for each smart city domain.

Setup of Flink Cluster:

Apache flink is a framework and distributed stream processing engine. Flink can scale horizontally on its cluster nodes. The first step before working with flink is to setup the flink cluster. In this experiment, multiple virtual machines with Ubuntu 16.04 are taken and flink is installed in them. Multi node flink cluster is been setup. Once the flink clsuster setup is done, we can see the information about the cluster like the health satatus of the cluster, worker node details, running, completed and failed jobs .etc through its portal. Setup is tested by verifying the portal. Sample flink application is submitted through its portal and made sure that it is working on the cluster as expected.

Integration of Flink and Twitter:

In our experiment, twitter will have tweets and we need them in flink environment to process them. Flink has a Twitter Source Connectors to bridge this gap. In this experiment, we used “TwitterSource” connector of flink to address this requirement. Sample example is run in the presence of internet and flink is able to read the tweets as expected.

Integration of Flink and Kafka:

Flink is a data processing platform. Kafka is a message broker. A big application can be designed as multiple flink jobs working together each gets its own input and processes it and produces its own output. There is need for a message broker to provide input and output to these flink jobs. Kafka is a good fit for this requirement. Flink have multiple source connectors whose purpose is to provide input to flink. Flink have multiple sink connectors to consume the flink output. Flink has “FlinkKafkaConsumer” and “FlinkKafkaProducer” for this purpose [9]. So flink can easily read from kafka topics and write to kafka topics. Sample flink job is done to read from kafka topic and to write to kafka topic and it workded as expected. Flink Kafka Conectors are used in this experiment.

Development of Flink Jobs:

Flink is basically a stream processing engine. It has a powerful Data Stream API which is useful to develop stream processing jobs. It is available in Java and Scala languages. Java is chosen for this experiment. Twitter messages are cosumed into flink environment using “TwitterSource” conncnetor of flink. The credentials obtained from “Twitter Developer Account” are specified in the program. Thus the tweets are consumed into flink and they are available as a flink data stream. Tweet is represented as a JSON file. It has multiple fields representing different types of information about that tweet. For example, “created_at”, “id”, “text”, “coordinates”, “hashtags” .etc. The fields that are important to this experiment is “coordinates” and “text”. These details are extracted from the tweet message. If the “coordinates” of the tweet are in the geographical range of samrt city, then that tweet is selected for further analysis. Otherwise it is droped. “text” field of the tweet represents the actual textual message of the tweet. That text content is extracted for the selected tweets. We have already discussed that set of keywords and phrases are formed for every vertical of smart city. For every domain, get it’s set of keywords, check for their availability in the text content of the tweet. If it contains any of those keywords, then treat that tweet is relevant to that respective smart city vertical and write that message to the respective kafka topic. Do this processing for every vertical of the smart city. Do it for every tweet that has the coordinates in the geographical range of smart city. Flink Jobs are developed to perform this work.

Deployment of Flink Jobs:

Once the flink jobs are developed, they are deployed on the flink cluster using flink portal. Flink portal will have an option to submit the jobs. It will have the provision to upload the jar files of flink jobs. Flink programs developed as above are exported as jar files. Those jar files are submitted to flink through its portal. Then the flink master will deploy them on the flink cluster.

Development of Web Technology to Stream the Kafka Messages to Respective Smart City Departments:

Flink jobs will read the tweets, filters them and write relevant messages to respective kafka topics. But the client needs those messages on his dashboard. To bridge this gap, we have chosen web technologies. Apache Tomcat is used as a web server.

At server side, servlet application is used to continuously read message in kafka topics. If some new message come to kafka topic, immediately it will be streamed to the browser. For that it is good to have a permanent http connection. For that we have choosen websockets. Once websocket is created between clent and server it will be used to send multiple messages. Whenever a new message comes to kafka topic, that will be sent to appropriate client through web socket.

Every vertical of smart city will have a webpage with dashboard. Consider that web page as client. The messages it received from the respective kafka topics will be visualized in it. First as explained above, client intiates the websocket connection. So client will be receiving the kafka messages relevant to its domain. Those messages are displayed on the webpage. The respective personnel observing those messages may take appropriate action.

Thus current application extracts the information regarding smart city service failures and displays that information on the department’s dashboard. With this application, it will be easy to monitor the service failures.

Conclusion

Smart city has multiple verticals like water management, garbage management, traffic management, power management .etc. It will be good to have an automated system to monitor the failure of smart city services. One way to build such system is to analyze the data coming from social networks because people will react to those service failures through their favorite social networks. In this paper, an application is designed and implemented to analyze the twitter data and find the tweets that are relavnt to smart city sevice failures and alert appropriate departments by sending those messages to take further action. Thus each smart city department will get the tweets that are talking about that service failure on their dashboard.

References

[1] MartÃn, Angel, Ana BelÃ©n Anquela JuliÃ¡n, and Fernando Cos-GayÃ³n. "Analysis of Twitter messages using big data tools to evaluate

and locate the activity in the city of Valencia (Spain)." Cities 86 (2019): 37-50.

[2] Suma, Sugimiyanto, Rashid Mehmood, and Aiiad Albeshri. "Automatic Detection and Validation of Smart City Events Using HPC and Apache Spark Platforms." Smart Infrastructure and Applications. Springer, Cham, 2019. 55-78.

[3] Souza, Arthur, et al. "Social smart city: A platform to analyze social streams in smart city initiatives." 2016 IEEE International Smart Cities Conference (ISC2). IEEE, 2016.

[4] Li, Mengdi, et al. "The new eye of smart city: novel citizen sentiment analysis in twitter." 2016 International Conference on Audio, Language and Image Processing (ICALIP). IEEE, 2016.

[5] Apache flink, Stream Processing framework available at: http://flink.apache.org

[6] Apache Kafka, Messaging Framework available at: http://kafka.apache.org

[7] Twitter, a Social Network available at: http://twitter.com

[8] Twitter Developer Account page available at http://developer.twitter.com

[9] Flink’s Kafka Connctor Explaination available at https://ci.apache.org/projects/flink/flink-docs-stable/dev/connectors/kafka.html