RELIANCE JIO MONEY SUMMER INTERNSHIP : PROJECT REPORT

Title : Exploratory Data Analysis for Jio Money Transit App

Submitted by

Prince Kumar
IIT Kanpur
3rd Year Undergraduate
Dated-12th July 2016

CONTENTS

ABSTRACT

During my internship at Reliance Jio Money I was able to experience the work of Software and Data Analysis Department of the Technical Company. This report contains what I have learned and done during the internship.

1 . Data Analysis

1.1 Task

- Collect the data using the API of the Operator.
- Merge the city list from two different sources (bus operators) with their Corresponding Id's on city names.

1.2 Challenges

- Spelling errors
- Old name and new names of the same city.

1.3 Tools Used:

- **Open Refine** is a tool for working on messy data.
  - This tool clusters the duplicate data using some string matching algorithms.
  - Algorithms used are Key collision method, N – gram fingerprint , Nearest neighbor method (Levenshtein Distance).
- **Pandas**
  - It is a python library for data manipulation and analysis .
  - DataFrame object for data manipulation with integrated indexing was mostly used.
- **Jupyter Notebook**
  - It is basically a web application more or less like python REPL, which was used for data cleaning and transformation along with Pandas .
- **Postman**
  - It is an Application which helps to build , test and document API faster.
  - In additional to testing API, the Generate code feature of this app was used to get the python request which had all the headers,which then was later used to get the data using http requests.

- **Boomerang**
  - It is an app that helped access and work with REST and SOAP web services.
- **Postgres**
- **Google Map API**
- **Bing Map API**

1.4 Approach

• Postman and Boomerang was used to test various API's of the Bus Operator

• After which python script was used to extract the data and dumped to Postgres on the local server .

• After extracting the data , Pandas and Jupyter Notebook was used to analyse and clean the data .

• Google and Bing Map Api was then used to get the latitude and the longitude of the cities

• At last open refine was used to remove the duplicacy in the data and some futher processing . and Pandas was used to merge the common cities .

Bitbucket link
https://bitbucket.org/princebgt28/intern_jiomoney16/data_analysis

2 . ELK (Elastic Search , Logstash , Kibana)

2.1 Task

- ∗ create a server that would listen to the user's log information and write the log on the port where logstash is listening.
- ∗ Display the data in the logstash graphically using kibana.

2.2 About

- **Elastic Search**
  - It is a search engine which can be used to search all kinds of documents .
  - It provides scalable search and has near real-time search, and supports multitenancy.
- **Logstash**
  - Logstash is a tool for managing events and logs. It wasused to collect logs, parse them, and store them in ElasticSearch.

- **Kibana**
  - This was used for data visualization based on the search results of the Elasticsearch
- **NodeJs**
  - Learnt about functional programming
  - It was used to to create the local server .

2.3 Approach

• Nodejs was used to create the local server which could get the log information of the user .

• **winston-logstash** library was used to send the log data to the logstash server over the tcp port.

• Logstash requires a config file which contains Input , filter and output section , the input section would listen on the same port on which nodejs server was writing log data.

• Finally the data was indexed and pushed to the Elasticsearch.

• Using Elasticsearch the logdata was queried and displayed visually on kibana.

Bitbucket link
https://bitbucket.org/princebgt28/intern_jiomoney16/nodejs_logstash

3 . Akka

3.1 Task

∗ make a node cluster scalable using Akka Actors.

3.2 About

- Akka is a toolkit and runtime for building highly concurrent, distributed, and resilient message-driven applications on the JVM .
- Akka is Responsive , Resilient , Elastic and Message driven or we can say it is simply a reactive system .
- In **Distributed system** every connected system communicated with each other to achieve common goal
- **Responsive** system responds in a timely manner if at all possible. This consistent behaviour in turn simplifies error handling, builds end user confidence, and encourages further interaction.
- **Resilient** system stays responsive in the face of failure.
- **Elastic** The system stays responsive under varying workload.
- **Message Driven** Reactive Systems rely on asynchronous message-passing to establish a boundary between components that ensures loose coupling, isolation and location transparency .

3.3 Approach

- Learnt Scala as the tutorial on web is mostly on scala and moreover it is easy to program in scala .
- Built a simple 2 Actor system that communicate with each other and terminates after 100 count is reached .

Bitbucket link https://bitbucket.org/princebgt28/intern_jiomoney16/akka