CSP-554-BIG DATA TECHNOLOGIES

PROJECT DRAFT

Location Based Restaurants Recommendation System

NOVEMBER 14, 2023

MANOGNA VADLAMUDI A20551908

ILLINOIS INSTITUTE OF TECHNOLOGY
PROF. PANCHAL J JAWAHAR

SECTION-I: INTRODUCTION

TOPIC: Location Based Restaurants Recommendation System

PROBLEM STATEMENT:

People are sharing their experiences on products or services in the form of reviews, which enables other users for their decision making. It is hard for the user to go through each review, and it might mislead the user to use a wrong service or a product, if he/she goes by the review given by a user with opposite taste. Therefore, we came up with a system which recommends the user depending on his needs. In the project only the positive part of the User Review was considered for sentiment analysis, which did not yield accurate results.

PROPOSED SOLUTION:

Our work proposed a new approach to recommend user better restaurants based on analyzing daily reviews. This helps the user to choose the better restaurant on the same day. we are able to extend working on the same by implementing sentimental analysis on negative User Review, which helps us to acquire much better results.

PROJECT GOALS:

- Recommend users better restaurants.
- Analyze daily reviews.
- Sentiment Analysis on positive and negative User Reviews.
- Predict restaurant ratings.

BIG DATA TECHNOLOGIES: Apache Kafka, Apache Spark, Elasticsearch
OTHER TECHNOLOGIES: PySpark, Scala, Apache Zookeeper, Python

The main objective of this project is to use big data technologies to build a location-based recommendation system that collects data and analyzes the data and recommend restaurants based on location in an optimized format such that implementable insights can be extracted from real-time data.

SECTION-II: LITERATURE REVIEW

II.I: INTRODUCTION:

These days reviews play vital role in every individual's life. It helps in choosing the best product or service. They have enhanced the probability of buying the right product, going to a good movie, eating at the best restaurant etc. But sometimes they could mislead a user in making a bad choice if he/she has opposite tastes when compared to the user who has written the review. So, to overcome this problem we have developed a system, which helps the user in choosing the right restaurant according to his/her requirements.

Our work comprises of two recommendation systems in which, one uses the Static data and other uses the real time streaming data. Both of these systems are discussed in the later sections of this paper. To withstand the high competition, restaurants try to improve their quality to satisfy their customers. We processed tips data on daily basis through streaming and recommended top restaurants to the user.

Apache Zookeeper, Apache Kafka servers are used to stream our static data, which are discussed in the later sections of this paper. Inverted index have been created in Elasticsearch and visualized using Kibana. Our systems were experimented on Yelp dataset and the results, future works are also discussed in the later sections. Java, Scala and Python were used for accomplishing the project.

II.II: Apache-Spark:

Apache Spark provides programmers with an application programming interface centered on a data structure called the resilient distributed dataset (RDD), a read-only multiset of data items distributed over a cluster of machines, that is maintained in a fault-tolerant way. It was developed in response to limitations in the MapReduce cluster computing paradigm, which forces a particular linear dataflow structure on distributed programs: MapReduce programs read input data from disk, map a function across the data, reduce the results of the map, and store reduction results on disk. Spark's RDDs function as a working set for distributed programs that offers a (deliberately) restricted form of distributed shared memory.

The availability of RDDs facilitates the implementation of both iterative algorithms, that visit their dataset multiple times in a loop, and interactive/exploratory data analysis, i.e., the repeated database-style querying of data. The latency of such applications (compared to Apache Hadoop, a popular MapReduce implementation) may be reduced by several orders of magnitude. Among the class of iterative algorithms are the training algorithms for machine learning systems, which formed the initial impetus for developing Apache Spark.

Apache Spark requires a cluster manager and a distributed storage system. For cluster management, Spark supports standalone (native Spark cluster), Hadoop YARN, or Apache Mesos. For distributed storage, Spark can interface with a wide variety, including Hadoop Distributed File System (HDFS), MapR File System (MapR- FS), Cassandra, OpenStack Swift, Amazon S3, Kudu,

or a custom solution can be implemented. Spark also supports a pseudo-distributed local mode, usually used only for development or testing purposes, where distributed storage is not required and the local file system can be used instead; in such a scenario, Spark is run on a single machine with one executor per CPU core.

II.III: Spark Streaming:

Spark Streaming leverages Spark Core's fast scheduling capability to perform streaming analytics. It ingests data in mini-batches and performs RDD transformations on those mini-batches of data. This design enables the same set of application code written for batch analytics to be used in streaming analytics, thus facilitating easy implementation of lambda architecture. However, this convenience comes with the penalty of latency equal to the mini-batch duration. Other streaming data engines that process event by event rather than in mini- batches include Storm and the streaming component of Flink. Spark Streaming has support built-in to consume from Kafka, Flume, Twitter, ZeroMQ, Kinesis, and TCP/IP sockets.

II.IV: Apache Kafka:

Apache Kafka is an open-source stream processing platform developed by the Apache Software Foundation written in Scala and Java. The project aims to provide a unified, high-throughput, low-latency platform for handling real-time data feeds. Its storage layer is essentially a "massively scalable pub/sub message queue architected as a distributed transaction log," making it highly valuable for enterprise infrastructures to process streaming data. Additionally, Kafka connects to external systems (for data import/export) via Kafka Connect and provides Kafka Streams, a Java stream processing library.

II.V: Apache Zookeeper :

ZooKeeper is a centralized service for maintaining configuration information, naming, providing distributed synchronization, and providing group services. All of these kinds of services are used in some form or another by distributed applications. Each time they are implemented there is a lot of work that goes into fixing the bugs and race conditions that are inevitable. Because of the difficulty of implementing these kinds of services, applications initially usually skimp on them, which make them brittle in the presence of change and difficult to manage. Even when done correctly, different implementations of these services lead to management complexity when the applications are deployed.

II.VI: Elasticsearch:

Elasticsearch is a search engine based on Lucene. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents. Elasticsearch is developed in Java and is released as open source under the terms of the Apache License. Official clients are available in Java, .NET (C#), Python, Groovy and many other languages. Elasticsearch is the most popular enterprise search engine followed by Apache Solr, also based on Lucene.

Elasticsearch is developed alongside a data-collection and log-parsing engine called Logstash, and an analytics and visualisation platform called Kibana. The three products are designed for use as an integrated solution, referred to as the "ELK stack".

SECTION-IV: REFERENCES

The list of sources below is a recommended reading list. The knowledge collection mechanism may determine whether or not a reference is eventually relevant to this program, and to what capacity/extent. As a result, all references listed below may or may not be referenced. Additional sources may also be applied during the review process.

[1] **https://github.com/patilankita79/Location-based-Restaurants-Recommendation-System/tree/master/BigDataProject**

[2] **https://developer.confluent.io/what-is-apache-kafka/**

[3] **https://haocai1992.github.io/data/science/2022/01/13/build-recommendation-system-using-scala-spark-and-hadoop.html**

[4] **https://docs.aws.amazon.com/lex/latest/dg/sentiment-analysis.html**

[5] **https://docs.aws.amazon.com/lex/latest/dg/sentiment-analysis.html**

[6] **https://pypi.org/project/zookeeper/**

[7] **https://www.yelp.com/dataset**