Report

Praxal Patel
Yash Thesia
Instructor : Prof. Goce Trajcevski
21 May 2018 - 6 July 2018

Preface and Acknowledgements :

For one and a half months from 20th May 2018 till 6th July 2018, we did a research internship at Iowa State University in the Electrical and Computer Engineering Department. This internship project is a part of our4- year bachelors program which we are pursuing at Nirma University, Ahmedabad, India.

We worked on a series of different assignments and projects. We learnt various machine learning techniques and usage by surveying research papers on trajectory embeddings and next point of interests. We also acquired skills to handle Big Data (Static as well as Stream Data) through Hadoop, Spark and StreamBase concepts.

Through the assignments, we did not only gain a lot of knowledge but more importantly, We also had a great chance to sharpen our skills in a supportive working environment.

We are very appreciative of Prof. Goce Trajcevski, our guide at Iowa State University. Mr. Goce gave us very in-time valuable instructions and put us in contact with his PhD student Xu Teng, who gave us extensive guidance regarding many practical issues. We also would like to express our gratitude to Arun Somani for having us here and providing us with all the required resources for our project.

Throughout the internship, we have also learnt many things about the culture here whose benefits are far beyond what we could learn in a normal project. In short, we would like to thank the Iowa State University Research scholars Internship Office for introducing us to this great opportunity in which we have developed ourselves both academically, professionally and socially.

1: Hadoop

Apache Hadoop is an open source software framework for storage and large-scale processing of data-sets on clusters of commodity hardware. Hadoop is an Apache top-level project being built and used by a global community of contributors and users.

The Apache Hadoop framework is different from the traditional relational database because of the following modules :
Hadoop Common: contains libraries and utilities needed by other Hadoop modules
Hadoop Distributed File System (HDFS): a distributed file-system that stores data on the commodity machines, providing very high aggregate bandwidth across the cluster
Hadoop YARN: a resource-management platform responsible for managing compute resources in clusters and using them for scheduling of users' applications
Hadoop MapReduce: a programming model for large-scale data processing.

1.1 Hadoop distributed file system

The Hadoop distributed file system (HDFS) is a distributed, scalable, and portable file-system written in Java for the Hadoop framework. Each node in a Hadoop instance typically has a single Namenode, and a cluster of Datanodes form the HDFS cluster. The situation is typical because each node does not require a Datanode to be present. Each Datanode serves up blocks of data over the network using a block protocol specific to HDFS. The file system uses the TCP/IP layer for communication. Clients use Remote procedure call (RPC) to communicate with each other.

An advantage of using HDFS is data awareness between the job tracker and task tracker. The job tracker schedules map or reduces jobs to task trackers with an awareness of the data location. For example, if node A contains data (x, y, z) and node B contains data (a, b, c), the job tracker schedules node B to perform map or reduce tasks on (a,b,c) and node A would be scheduled to perform map or reduce tasks on (x,y,z). This reduces the amount of traffic that goes over the network and prevents unnecessary data transfer. When Hadoop is used with other file systems, this advantage is not always available. This can have a significant impact on job-completion times, which has been demonstrated when running data-intensive jobs. HDFS was designed for mostly immutable files and may not be suitable for systems requiring concurrent write-operations.

Another limitation of HDFS is that it cannot be mounted directly by an existing operating system. Getting data into and out of the HDFS file system, an action that often needs to be performed before and after executing a job can be inconvenient. A filesystem in Userspace (FUSE) virtual file system has been developed to address this problem, at least for Linux and some other Unix systems.

File access can be achieved through the native Java API, the Thrift API, to generate a client in the language of the users' choosing (C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, Smalltalk, or OCaml), the command-line interface, or browsed through the HDFS-UWe web app over HTTP.

1.2 JobTracker and TaskTracker: The MapReduce engine

Jobs and tasks in Hadoop

Above the file systems comes the MapReduce engine, which consists of one JobTracker, to which client applications submit MapReduce jobs. The JobTracker pushes work out to available TaskTracker nodes in the cluster, striving to keep the work as close to the data as possible.

With a rack-aware file system, the JobTracker knows which node contains the data, and which other machines are nearby. If the work cannot be hosted on the actual node where the data resides, priority is given to nodes in the same rack. This reduces network traffic on the main backbone network.

If a TaskTracker fails or times out, that part of the job is rescheduled. The TaskTracker on each node spawns off a separate Java Virtual Machine process to prevent the TaskTracker itself from failing if the running job crashes the JVM. A heartbeat is sent from the TaskTracker to the JobTracker every few minutes to check its status. The Job Tracker and TaskTracker status and information is exposed by Jetty and can be viewed from a web browser.

1.3 Projects in Hadoop :

a. Word Count in Hadoop
b. Finding Average of given data.
c.Finding frequency of triangles in an undirected and directed graph.

2: Apache Spark

Apache Spark is a fast, in-memory data processing engine with elegant and expressive development APIs to allow data workers to efficiently execute streaming, machine learning or SQL workloads that require fast iterative access to datasets.

The Hadoop YARN-based architecture provides the foundation that enables Spark and other applications to share a common cluster and dataset while ensuring consistent levels of service and response.

Spark is designed for data science and its abstraction makes data science easier. Data scientists commonly use machine learning – a set of techniques and algorithms that can learn from data. These algorithms are often iterative, and Spark's ability to cache the dataset in memory greatly speeds up such iterative data processing, making Spark an ideal processing engine for implementing such algorithms.

The knowledge about Spark will also helpusto handle data for various machine learning projects which We intend to undertake in the future.

3: Stream Processing

Stream Processing is a Big data technology that enables users to query continuous data stream and detect conditions fast within a small time period from the time of receiving the data. The detection time period may vary from few milliseconds to minutes. For example, with stream processing, you can receive an alert by querying a data stream coming from a temperature sensor and detecting when the temperature has reached the freezing point.

Stream processing naturally fit with time series data and detecting patterns over time. For example, if you are trying to detect the length of a web session in a never-ending stream (this is an example of trying to detect a sequence). It is very hard to do it with batches as some session will fall into two batches. Stream processing can handle this easily.

Batch processing lets the data build up and try to process them at once while stream processing process data as they come in hence spread the processing over time. Hence stream processing can work with a lot less hardware than batch processing. Furthermore, stream processing also enables approximate query processing via systematic load shedding. Hence stream processing fits naturally into use cases where approximate answers are sufficient.

Finally, there are a lot of streaming data available (e.g. customer transactions, activities, website visits) and they will grow faster with IoT use cases (all kind of sensors). Streaming is a much more natural model to think about and program those use cases.

Software Used: Tibco StreamBase

Project: Real-Time Stock Analysis

Chapter 4: Assistance in survey of Location-Awareness in Time Series Compression.

4.1 Finding if a given latitude and longitude is falling inside the United States using a dataset having boundary coordinates of all the countries.

4.2 Comparative analysis by plotting bar graphs of Similarity Score vs Location.

5. Summarising Identifying Human Mobility via Trajectory Embeddings

In this study, the authors have used Trajectory User Linking via Embedding and RNN for identifying human mobility. The users of Location Based Social Networks are taken into consideration and their locations are derived using the geospatial tags and taken as data.
The motivation of TUL (method proposed in this paper)was mainly due to the reason that the traditional applications focussed on modes of transportation and activity patterns of the users by using Dynamic Bayesian Network, Hidden Markov Model and Conditional Random Field.

Linking trajectories generated by the user can help make better recommendations. (for eg. Uber generates information about the trajectories of the user and it can be utilised by some bike/car sharing applications.

Problems in TUL :
1. A large number of mobile users.
2. Sparse Trajectory Data as well as noise and outliers.
3. User privacy invasion and greater dimensionality because of the many features.

The dataset used is Gowalla(201 users) and Brightkite (92 users).

The first step taken was to link the unlinked trajectories with its corresponding users. For the same, the check-in embeddings (a. follows the power law distribution curve, b. addresses the overfitting problem) and linked user trajectories are fed to the RNN model for predicting the user of the unlinked trajectory in question.
The embedding of a check-in includes predicting its probability in context to a particular location. To reduce the problem of higher dimensionality, we have used a Check-in Matrix.The study has used a given sized sliding window and log of probability is calculated. For optimisation, a dropout layer is included where we randomly drop some check-ins and the corresponding rows in the matrix are set to zero.

Various variations of RNN like TULER-LSTM, TULER-GRU, Bi-Directional TULER , Longest Common Subsequence, LDA and SVM have been used for the experiment.
Out of them, TULER-LSTM faired better for the Brightkite Dataset and Bi-TULER for the Gowalla Dataset. The performance was evaluated using Macro-F1 and ACC@1 and ACC@5 measures.

During the experiments, it was concluded that by increasing the number of iterations and having a smaller learning rate, higher accuracy can be achieved.

This approach has its drawbacks when the check-ins are very sparse and thus we will need to include extra attributes like timestamps.

The performance can be improved even further if we use more and more social network information and community moving patterns.

6. Summary

The overall experience at the Iowa State University was one we would cherish. The working environment being one of the best ones we have come across. This has truly been a great learning experience and we'll be forever indebted to those who gave us a hand here.