Project Report
Mining Social Network Data using the Hadoop/MapReduce Framework

Abhisht Alkargud Sathyanarayana
abhishtas@uci.edu
Karan Kaushik
kkaushik@uci.edu

March 21, 2011

1 Introduction

This report describes our project which extracts data from social networks and performs computation on the data using a distributed service. This data is stored on a distributed file system. A custom MapReduce job is written to perform specific processing on the data. The data is then visualized in the form of a graph. The goal of this project is to present a broad overview of the current technologies surrounding distributed file systems, social network data mining and formulating problems using the MapReduce framework. We present a brief description of the modules used in the project.

2 Overview

Design

The project consists of four components, the first module utilizes a twitter API for python to grab relevant data and write it to disk. In this demonstration we have extracted entities and tweet id’s of each tweet for a particular user and written it to a file. The program takes care not to exceed the resources of the server by restricting its extraction rate and sleeping for some time when it exceeds the allowed rate. We must also handle a three way authentication protocol to authenticate with the twitter API and store the public key in our system. The second module is the MapReduce operation. It takes as an input the list of tweet id’s and corresponding entities. It then counts the frequency of the entities. The map operation emits entity, for each occurrence of the entity and the reduce operation aggregates over these to
return a list of frequencies of each entity. The third module is used to visualize the output, It takes as an input the output of the MapReduce operation and generates a graph.

3 MapReduce

MapReduce using Hadoop streaming

MapReduce is a framework, that supports distributed computing on large amounts of unstructured data on a cluster of commodity computers. This framework is provided with two functionalities called ”Map” and ”Reduce”. The map function takes in key/value pairs as input to generate a set of intermediate key/value pairs. The reduce function is then applied on this set, which helps to merge all the intermediate values associated with the same intermediate key. The resource utilization is made seamless by having a run-time system take care of details of partitioning the input data, scheduling execution over different machines, handling failures and the necessary communication overhead. The necessity for the usage of MapReduce for our project arose out of the need to process a potentially large data set.

Hadoop Streaming

Usually MapReduce operations for Hadoop are written in Java. Hadoop has a provision by which we can write our MapReduce modules in any programming language of choice which can read and write to stdin and stdout in Unix. Python was thus our language of choice as it has a vast amount of libraries for text processing and easy to use API’s to extract data from a variety of data sources and RESTful apis provided online. We thus construct two python modules to perform the Map and Reduce operation and successfully store the output in the Hadoop Distributed File System(HDFS).

The Reduce Operation

A machine chosen to be a reduce worker is then notified by the master about these locations and these then read the intermediate data. By sorting all the intermediate keys, all similar occurrences are then grouped together. This is then processed through the user defined reduce function and this result is appended to the final output. These R files that are obtained as output can then be used for other distributed applications. The master maintains several
data structures like idle, in-progress or completed for Map and Reduce task and also identity of the worker machine. As for faults in the system, there can be different types of failures- Worker failure and Master failure. Failures as mentioned above are handled only by using re-execution of the operation at the node or the whole map reduce operation depending on whether the failure was that of a worker or a master machine. For a map function, the machine writes the R files to a temporary file. A reduce function write to one final output file and lets the master know that the operation has been completed. A common problem that is encountered in most mapreduce operations is a straggler. Stragglers are machines that take longer time to complete the last few steps of a given mapreduce operation. To handle this, the master schedules backup executions of the remaining in-progress tasks. A task is said to be complete if execution of the operation is completed either in the primary or the back up.

4 Distributed File Systems

Some of the prior work in the field has been focused on building distributed file systems for specific application to web crawlers. The Hadoop Distributed File System is a distributed file system designed to run on commodity hardware. It allows the capacity of a cluster to be expanded by scaling out instead of scaling up. It is designed from the ground up with fault tolerance in mind. Some of the assumptions and design considerations are inspired by the Google File System. It assumes that constant failures are the norm and that data access is usually streaming. Only a limited subset of the POSIX semantics are implemented. The system is designed for more throughput while accessing large data sets. It provides redundancy of data to deal with excessive node failure. It also acts as a distributed data store, where the processing is pushed to the data rather than other way around.

5 Implementation

The first step in our approach makes us deal with around 3200 tweets of a user. We first obtain the contents of his tweet in JSON Object format. This format helps us by making the initial text processing easier and lets us directly access the entities which are importance to us. The entity that we have focussed on is the hashtags. The field “id” gives us the identification number of that user. The structure of one such obtained result is as below.
"Of course I'm baiting a penguin?"
Using this "Id" and the various hashtags for one more users, we can do various types of computations. Our outputs indicate that words like "in" and "hadoop" are terms that have been most frequently tweeted by this user with frequencies of 121 and 54 respectively. The graph plots only 12 terms which have been sorted based on their frequencies in the users last 3200 tweets. Such a tool provides us with the ability to analyze what the user may be thinking or working on and this can be used as a tool to direct relevant ads towards them or even use the data as a reference to a greater set which could house such data of many users.

From a gestalt of all this data we could make a guess of what are the currently trending topics.
From the above data it is possible to generate a list of users who are tweeting the trending terms. With access to geo-data exposed by the Twitter API we can now map the user’s location on to a map using the Google Maps API and directly visualize the trend. The purpose and benefits of doing such an operation is we can let other users know about who else may share a common interest with their ideas and hence help them get connected. Another solution this provides is to try and seek out a cluster and to be able direct relevant ads to that region and improve the quality of penetration of that advertisement in that area.
From the above two figures we try to find the users who have tweeted about different topics in different regions of the world. Using the geo-codes of the users from the two areas we can perform the required analysis and try and determine trends and be able to visualize clusters of users who are tweeting about the same topic or any other similar activity.

6 Applications of Mining Data from Social Networks

Recently, there has been tremendous interest in the phenomenon of influence propagation in social networks. Studies in this area assume an input to problems to be a social graph with edges labeled with probabilities of influence between users. The collection of the user generated content data to calculate these probabilities come from computations from real social network data which was largely ignored until now. Thus, from a social graph and a log of actions by its users, one can build different models of influence. In addition to proposing models and algorithms for learning the model parameters and for testing the learned models to make predictions, they have also developed techniques for predicting the time by which a user may be expected to perform an action. Apart from the accuracy of their prediction, their research shows that it is vital to tap the vast data that is available from the Social Media market. One of the best sites to mine for user data is Twitter. A compelling reasons for mining Twitter data is to try to answer the question of what people may be thinking of at any given moment or right now. By conducting simple analysis of tweets, for example the most frequently used term in the tweets, we can deterministically state a user's intentions.

7 Future Work and Improvements

The future of this work, in true spirit of distributed computing would be to scale to even larger datasets. The sole media that has been used in this project is text based. With the available data we have been able to create an index, sort the words, rank the words and draw a frequency graph of the words thus obtained. The map feature also provides us with details of how certain terms are being echoed by different users who may be geographically seperated. The advancements possible in this regard are better string processing techniques to try and improve the quality of the generated
outputs. Stemming and removal of stop words would then play a vital role in trying to get only the unique terms from tweets and hence refine our results when looking over a larger dataset. Additional features to this setup could be the use of multimedia rather than just plain text. The computation required to be able to retrieve an image from a large database is enormous and not feasible if run on just one node. If this computation could be done across different nodes, in a distributed fashion, we could achieve results in a much more speedy manner. One such implementation could be to use object recognition in images and to be able to sort and keep them with regard to some sort of ranking algorithm. This could be vital especially in cases like identification during natural disasters in which case the only way to make any judgement on a person’s identity would be to try and virtually recognize the person based on the available content already in existence, and use some sorting algorithm or face recognition software to try and judge which image could be the closest match and give a ranked list of potential candidates for the same. The computation for face is very computationally expensive and hence running it in a distributed manner thus allows achieve the fastest and the best results. The attempt at trying to make everything a part of an ubiquitous system, leads to use of ideas like voice commands and speech recognition. These applications will require us to go through a vast number of words and try to match them accurately to a given voice input. Again this is something that is very computationally intensive and can be best handled by modeling the problem using MapReduce on a distributed framework like Hadoop.