CprE 419 Lab 5: Analyzing Twitter Data using MapReduce

Department of Electrical and Computer Engineering
Iowa State University
Spring 2014

Purpose

In this lab, you will use Hadoop MapReduce for analyzing data from Twitter. The data from twitter consists of a number of messages or "tweets". Each tweet, in addition to text, can also contain a "hashtag", which can be thought of as a topic to which a tweet can belong. In addition, a tweet from one user can be "retweeted" by another user, and this can also be inferred by looking at the fields within the tweet.

In this lab, you are required to analyze the data to find out data about popular hashtags and retweets from users. We have constructed a dataset of actual tweets from twitter. You are required to identify information about hashtags and users from this dataset.

Submission

Create a zip (or tar) archive with the following and hand it in through blackboard**. Failure to include all elements will result in a loss of points.**

- A write-up answering questions for each experiment in the lab. For output, cut and paste results from your terminal and summarize when necessary.
- Commented Code for your program. Include all source files needed for compilation.

Dataset

We have collected a dataset of tweets using the Twitter API, centered around the hashtag "#oscars" and "#usa". The data is in the JSON format. This means you will have to do some more advanced handling of the data.
The dataset is located at: **"/class/s15419x/lab5/oscars.json" and "/class/s15419x/lab5/usa.json".**

JSON

For the next three experiments you will be parsing JSON-formatted strings. JSN is in its most simple explanation, a lightweight data-interchange format. It is commonly used in web applications to package data.

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.

An object is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right brace). Each name is followed by : (colon) and the name/value pairs are separated by ','
An array is an ordered collection of values. An array begins with [(left bracket) and ends with] (right bracket). Values are separated by ','
Note that the dataset is an array of JSON objects.

Examples of JSON:

```
[
	{
		"name": "John Smith",
		"salary": 1000,
		"hobby": "Bob-sledding"
	},
	{
		"name": "Paul Bunyan",
		"salary": 13200,
		"hobby": "Cricket"
	},
	{
		"name": "Daisy",
		"salary": 10000,
		"hobby": "Hockey"
	},
]
```

This is an array of objects with each object having an attribute of “name”, “salary”, and “hobby”. If you have any more questions on JSON, there is a more complete explanation at http://json.org/

This means you can take the string and turn it directly into a JSON object. There are many third party JSON parsers you can use to do this; we encourage using GSON or simple-json. We have uploaded the jar files for each of these libraries on piazza.

Resource:

- GSON: https://code.google.com/p/google-gson/
- Simple-json: https://code.google.com/p/json-simple/

Up until now we have used the textInputFormat format to read in data from HDFS. The next part of this lab requires the use of a CustomInputFormat and a CustomRecordReader. First let’s talk about what an input format and a recordReader does.
The input format specifies how we want to split our data into **chunks**. Each **chunk** is sent to a different mapper. We can change the logic for how we break up the data into chunks so we can send complete json-formatted strings to a mapper. Once in a mapper the recordReader breaks the **chunk** into **records**. Each record is a single key, value pair. A chunk has many records in it. The splitting of a file into chunks is handled by the inputFormat which logically breaks the file into chunks based on your custom recordReader.

We can use the recordReader to read in a single JSON object. Once done, it will read through a chunk until it finds the end of the JSON object. To do this we need to define the bounds of a JSON object. In JSON, each object starts with a ‘{‘ and ends with a ‘}’. You can ignore any characters outside of a JSON object (ie, a ‘,’ or ‘[‘ or ‘]’) Therefore, a good approach for writing code to recognize a JSON object would be to read starting at a ‘{‘ and reading until we find the matching ‘}’. We can write this new logic in a custom record reader. The concepts and some example code are explained well here: http://hadoopi.wordpress.com/2013/05/31/custom-recordreader-processing-string-pattern-delimited-records/

For our record reader we need to implement 6 different methods:
Initialize(): Prepares the record reader to start reading
nextKeyValue(): Creates the next Key Value pair. It will return true until it reaches the end of the file.
getCurrentKey(): Returns the current key
getCurrentValue(): Returns the current value
getProgress(): Reports how far into the file we have read
close(): Used by the framework for cleanup.

Of these functions the ones we will have to drastically change the logic of are initialize and nextKeyValue.
Note that in order to complete this experiment you will have to write a custom inputFormat(which implies writing a custom recordReader).

Experiment 1 (20 pts)
Find the top ten most common hashtags in the dataset, oscars.json. If a hashtag appears twice in the same tweet, only count the hashtag once. Note that many tweets will have the same text because one user retweeted the tweet. This does not count as a duplicate. Include the results in your report.

Experiment 2 (20 pts)
Find the top ten most followed tweeters in the dataset, oscar.json. Output the name and number of followers for each tweeter. If you look in the JSON input file example you will see a user field. This field describes the tweeter in more detail. Here there is a "screen_name" field and a "follower_count" field. These are the fields you should use for your analysis. Remember that a user can have different follower counts across different tweets. Choose the highest occurring follower count for each tweeter. Include the results in your report.

Experiment 3 (30 pts)
For each of the tweeters in the top ten most prolific tweeters find the most commonly used hashtag for each tweeter that is not #usa in **usa.json**. Prolific means the most number of posts, not the most number of followers. We want to see what these prolific tweeters talk about the most. This experiment will be more difficult as you will need to keep tweet information together. Include the results in your report.