UPPSALA
UNIVERSITET

Large Datasets for Scientific Applications

Assignment 1

Hamza Imran Saeed

April 26, 2019

Part I: Intro to HDFS/Hadoop and Wordcount

Task 1.1 (Word Count Example)

1. Look at the contents of the folder "output" - what are the files place in there? What do they mean?

There are 2 files placed in the output folder, `_SUCCESS` and `part-r-00000`. The `_SUCCESS` file is created by the MapReduce framework in the output directory so that applications can check if a job has been completed by looking at the Hadoop filesystem (though the file was created in the local file system in this scenario). The `part-r-00000` file contains the output of the job. The name of the file tells us whether it was a map job "-m-" or a reduce job "-r-" and the number that follows this is the mapper or reducer task number.

2. In this example we used Hadoop in "Local (Standalone) Mode". What is the difference between this mode and the Pseudo-distributed mode?

In the `Local Standalone Mode` (which is the default made in which Hadoop runs) Hadoop runs in a non-distributed mode, as a single Java process. HDFS is not used and instead the local file system is used for input and output.

The `Pseudo-distributed Mode` is like a single-node cluster where different types of nodes such as the Name Node and Data Node are running on the same machine. Each of the different nodes (hadoop daemons) run as different JVM instances. The instances communicate across network sockets and produce an optimized cluster on a single host. To handle this communication and running hadoop in `Pseudo-distributed Mode`, configuration files such as `core-site.xml` and `hdfs-site.xml` are used.

Task 1.2

1. What are the roles of the files core-site.xml and hdfs-site.xml?

The `core-site.xml` and `hdfs-site.xml` files are configuration files for Hadoop. The `core-site.xml` file defines properties such as port number, memory, memory limits, size of read/write buffers used by Hadoop. Similarly, the `hdfs-site.xml` defines the main configuration properties for the HDFS such as the paths of the Name Node and Data Node as well as replication factor.

2. Describe briefly the roles of the different services listed when executing 'jps'.

The services are listed and briefly described below:

- **NameNode**: controls the HDFS file system. It keeps the directory tree of all files in the HDFS, and tracks where in cluster the file data is kept. The NameNode communicates with client applications and returns list of relevant DataNodes when prompted for data.
- **DataNode**: stores data in the HDFS. A functional hadoop filesystem usually has more than one DataNode, with data replicated across them. The DataNode connects to the NameNode and responds to requests for filesystem operation by the NameNode or client applications.
- **SecondaryNameNode**: dedicated node in HDFS cluster that makes checkpoints of the file system metadata present on NameNode. This is not a backup NamenNode and cannot be used to replace the main NameNode. It is just a helper which stores the `FSImage` and `edits` log file.
- **JPS**: Java Virtual Machine Process Status Tool. Used to check what hadoop services are running on the machine.

Task 1.3

1. Explain the roles of the different classes in the file WordCount.java.

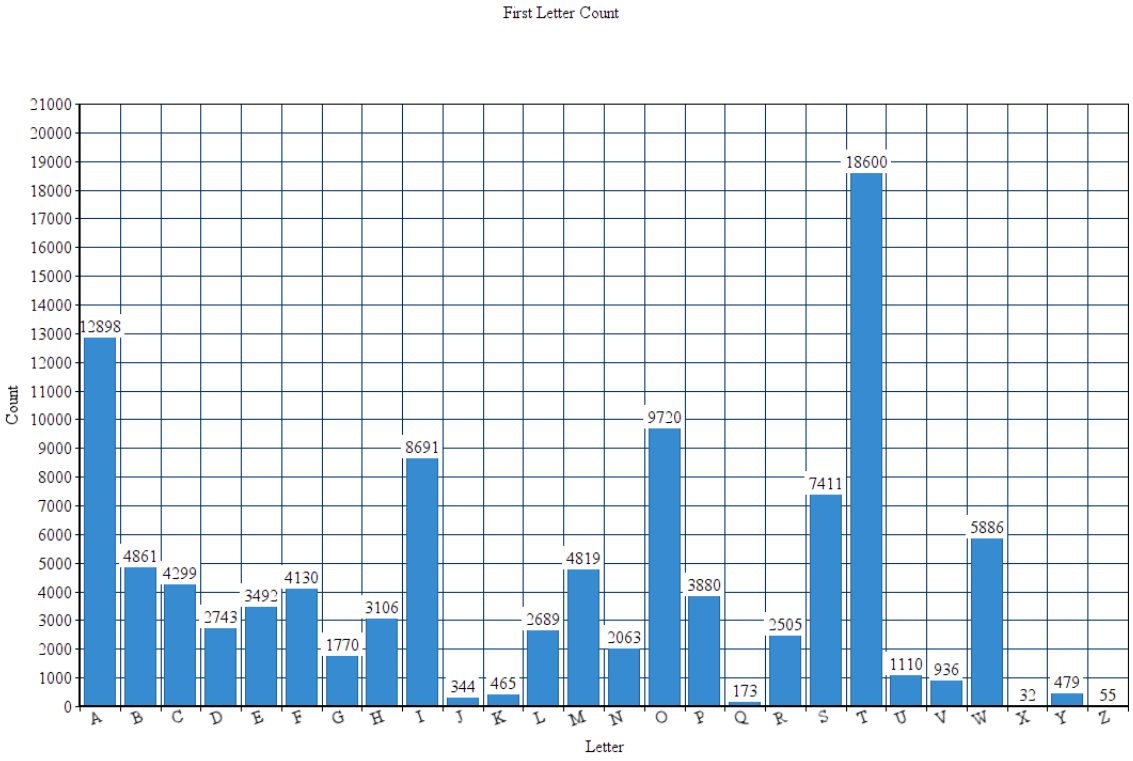
There are 3 classes in the file `WordCount.java`. These are `WordCount`, `Map` and `Reduce`. The `WordCount` Class is the main class which has the `main` method that is used to run the program. In the `WordCount` Class there are two more Classes, `Map` and `Reduce`. The `Map` Class takes a line from the Input file, breaks it down into words and then outputs to the `OutputCollector` the word paired with a 1 e.g (word,1). The `Reduce` Class takes the key-value pairs provided by the mapper and sums all the values for similar keys and then collects them for output.

2. What is HDFS, and how is it different from the local filesystem on your virtual machine?

The HDFS (Hadoop Distributed File System) is a file system for storing large files on a distributed cluster of machines. The HDFS is a virtual file system on top of the local file system. It stores data in blocks (similar to the local file system but the block sizes are larger) which are replicated and distributed across the cluster.

Task 1.4

1. Make a plot of some sort that shows the counts of each letter and include that in your report. Do not include the entire output file.

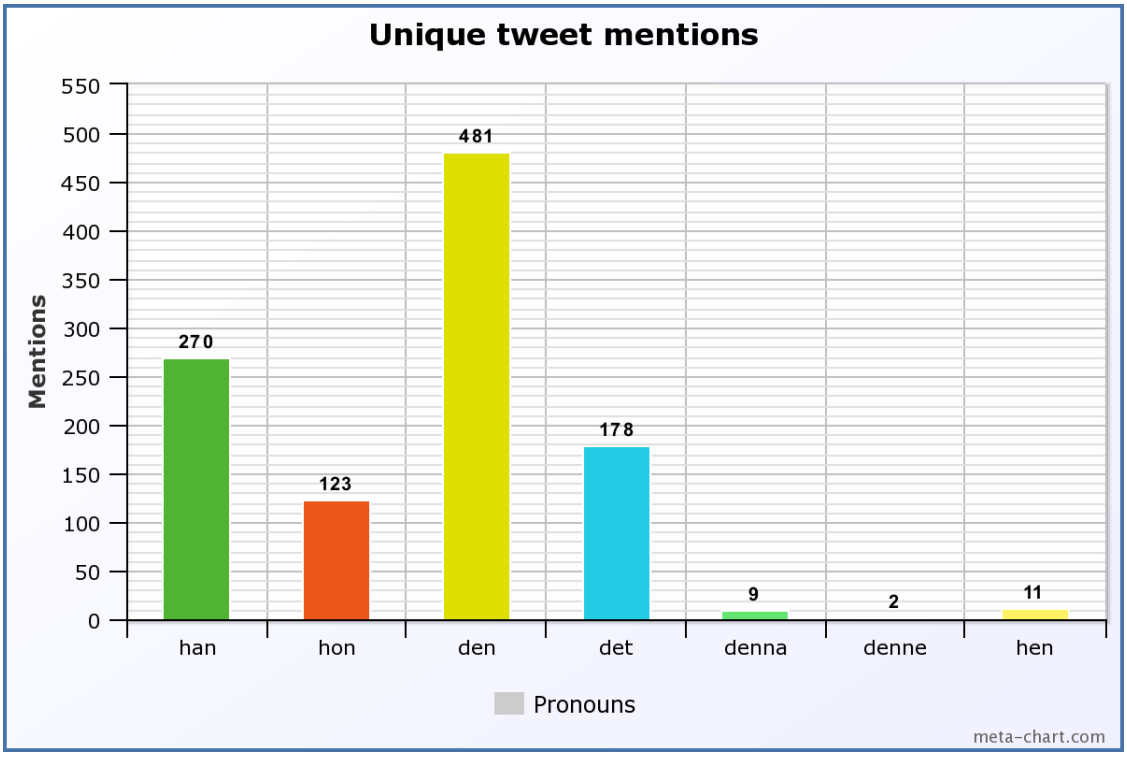


Part II: Analyzing twitter data using Hadoop streaming and Python

1. Based on the documentation in the above link, how would you classify the JSON-formatted tweets? Structured, semi-structured or unstructured data? What could be the challenges of using traditional row-based RDBMs to store and analyze this dataset (apart from the possibility of very large datasets)?

Based on the documentation, the JSON-formatted tweet data is semi-structured because although the tweets have a similar structure (because different attributes are used by different application for different purposes) all the tweets do not have the same structure e.g unique tweets are different from re-tweets because of the `retweeted_status` attribute. Traditional RDBMs have inflexible schemas and do not work with semi-structured data as they require structured data. Working with and RDBMs could require time to be spent on pre-processing of data to be loaded into the database(instead of just focusing on the analysis). Anytime that twitter makes a change to the api, the RDBMs (or atleast the script or process used for inserting the data) has to be adapted accordingly.

2. Your task is now to use Python and the Hadoop streaming API to count the number of tweets mentioning each of these pronouns and plot a bar chart of the counts,normalized by the total number of unique tweets (e.g counts-"hen"/total-unique)



The output from running the program is shown below, normalized against the number of unique tweets:

```
TOTAL   3215575
UNIQUE  2341577
hen     25508/2341577   = 0.0109
denne   3678/2341577    = 0.0016
den     1125508/2341577= 0.4807
denna   20934/2341577   = 0.0089
hon     287734/2341577  = 0.1229
han     631961/2341577  = 0.2699
det     415773/2341577  = 0.1776
```

The counts are normalized against the total number of unique tweets(tweets that don't have a `retweeted_status`) as shown above. The result is then multiplied by 1000 and rounded up to get integer values for comparison in the graph above.

Part III: High-level interfaces/tools on top of the Hadoop framework

Hive Queries

The query used to create the Hive table for storing the tweets is listed below. Only a subset of the json object keys were used as table columns because not all the columns were required to perform the required analysis. The main columns required were `id`, `text` and `retweeted_status`.

```
CREATE EXTERNAL TABLE tweets ( id BIGINT, created_at STRING, source STRING,
favorited BOOLEAN, retweeted_status STRING, text STRING) ROW FORMAT SERDE
'org.openx.data.jsonserde.JsonSerDe' STORED AS TEXTFILE;
```

To load the data into the table, the following query was used.

```
LOAD DATA INPATH '/tweets/files/' INTO TABLE tweets;
```

Finally, the query used to extract the results was:

```
SELECT word, COUNT(*) FROM tweets LATERAL VIEW
explode(split("han hon den det denna denne hen", ' '))
lateralTable AS word WHERE array_contains(split(lower(text),' '), word)
AND retweeted_status is NULL AND id is not NULL GROUP BY word;
```

Using this query the result is listed below which matches the result obtained from running MapReduce using python.

```
+--------+----------+
|  word  |   _c1    |
+--------+----------+
| hen    | 25508    |
| denne  | 3678     |
| den    | 1125508  |
| denna  | 20934    |
| hon    | 287734   |
| han    | 631961   |
| det    | 415773   |
+--------+----------+
```

1. Briefly discuss and compare the experiences of using Hive vs. "vanilla" Hadoop/MapReduce and Hadoop streaming. Focus on the user experience. Since we are working on a toy-sized dataset, performance comparisons become a bit shady, but as long as you are aware of that you can still measure and report it.

The user experience when using Hive vs "vanilla" Hadoop/MapReduce and Hadoop streaming is different and there are pros and cons in both techniques. Using Hive, the user is able to focus more on the task at hand and try to write an optimized query which gives a the specific result. The user does not have to worry about how to write a mapper and reducer separately. The user can also interact with the data for example get the text of the first 5 tweets or the last 5 tweets as you would in a SQL based RDBMs and this can be really helpful in terms of understanding the data and making decisions about how to write the query to get a desired result (i.e how to approach the problem). If the user has a background of working with traditional database technology, Hive can be easier to understand and work with. Because the level of abstraction is higher in Hive, the user can focus only on the task. On the other hand "vanilla" MapReduce and Hadoop Streaming give the user more control and users can write more complicated business logic. Hive functions have their limitations for example in this part of the assignment while writing the hive query, I used a Lateral Table in the query and split the text attribute

of the tweet into an array. I needed to get unique values from the array so that the same word was not counted twice but Hive did not allow functions such as `collect_set` to run on that part of the query (after the Lateral Table statement) and I had to re-write the query in a different way to get the program to count it as a single mention even if the word occurred in a tweet more than once. In python ("vanilla" Mapreduce), such a scenario would have been easily managed. Using "vanilla" MapReduce, a user can optimize the code and get a better performance from the framework. Hive requires the data to be structured and loaded into tables for querying and that requires knowledge of the data whereas in "vanilla" MapReduce the data doesn't have to be structured so in this sense "vanilla" MapReduce has a better user experience. Another difference that was noted during the given assignment, in terms of user experience, was the fact that if for example an error was made in the reducer script, the whole mapper part of the job ran and when the reducer part started, the program gave an error and job failed whereas when using hive, because the mapper and reducer are not provided by the user, this did not happen and queries that had errors or syntax issues were highlighted before running a significant part of the job. In terms of performance, the Hive query was observed to run slower than the python MapReduce code.

2. Briefly discuss, contrast and compare Pig vs. Hive. Note that you are not required to repeat the task using Pig.

Hive was created by Facebook and uses HieQL which is a declarative language based on SQL whereas Pig was created by Yahoo and uses Pig Latin as the programming language which is a procedural data flow language. Hive is used with structured data whereas Pig is used for structured and semi-structured data. Hive is similar to SQL as already mentioned and the user has to define the tables beforehand and store the schema details in any local database whereas in case of Pig there is no dedicated metadata database and the schemas or data types are defined in the script itself. Hive Hadoop runs on the server side of the cluster whereas Pig runs on the client side of the cluster.

3. In this assignment, the Twitter dataset was quite modest in size, only 10GB of raw JSON data. Here, we could even load it in main memory on a high-end workstation. For sake of argument, assume that it had been 10 times as large, say 100GB, it would still not have been a very big dataset, but too large to manage in a naive way. During Lectures 2 and Lectures 3 you heard about other types of data management systems supporting analysis. Which of these tools could have been efficient and productive for analyzing this specific dataset?

If the data was large (100 GB) and given the fact that the data is in the form of tweets(semi structured json objects), other data management systems such as MongoDB and Cassandra and other type of NoSQL databases would have been a good tool to use because of the fact that they are distributed and can be scaled horizontally. Furthermore, tools like SciDB can also be used which does not have strict schema dependencies and is highly optimised for multidimesnional datasets.Using Openstack Swift storage could also be used to store and analyse the data efficiently.