UPPSALA
UNIVERSITET

Data Engineering I - Assignment 2

Alexis Tubulekas

17 februari 2021

Task 1.1

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

_**SUCCESS**: To indicate that a successful job in Hadoop has been performed this file is created.

part-r-00000: The output from the actual WordCount performed on the text file pg20417.txt. The 'r' indicates that the job contained a reduce and '00000' is the reducer task number. The output is in the form of tab separated key-value pairs with the key being the word and the value being the number of times it appears in the text corpus.

b. How many times did the word 'Discovery' (case-sensitive) appear in the text you analyzed?

'Discovery' appears 5 times in the text file pg20417.txt

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

By default Hadoop runs in local standalone mode, as a single Java process in a single Java Virtual Machine. HDFS (Hadoop Distributed File System) is not used in this mode. This mode is mainly used for the purpose of learning, testing and debugging.

In Pseudo-distributed mode Hadoop utilizes HDFS (Hadoop Distributed File System) and behaves like a real distributed cluster, but in fact it only uses one single node meaning the cluster is simulated, thus the processes occur independently.

Task 1.2

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

Hadoop stores its configuration in XML files.

core-site.xml: Contains information about where the NameNode runs in the cluster.

hdfs-site.xml: Contains the configuration settings for the HDFS services: the NameNode, SecondaryNameNode and DataNode. It is also where it is configured how many times the data should be replicated, which is done for fault tolerance.

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

NameNode : Responsible for tracking where in the cluster the data files are located (on what DataNodes). It does not store the data itself. Applications communicate with the NameNode to locate a file.

Secondary NameNode: Optional NameNode that can be hosted on a different machine. It is not a backup for the NameNode as it does not provide real redundancy as it is limited to checkpoints.

DataNode: DataNodes stores the data in the HDFS. There usually exists multiple DataNodes with the data replicated among them. A typical replication factor is 3. It is in the DataNodes that run the actual computations, so the computations occure close to the data.

Task 1.3

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

Class Map: This is the mapper. It takes in a line from the text file and converts it to string type and splits the line into words. Then it iterates through every word and forms key value pairs for each word as <word,one> and pushes to the output.

Class Reduce: This is the reducer. It loops through the output of the class map, checks if the next word is the same as the current and if so adds the value until the next word is a different word, then it outputs the key (the word) and the value i.e. how many times it occurred.

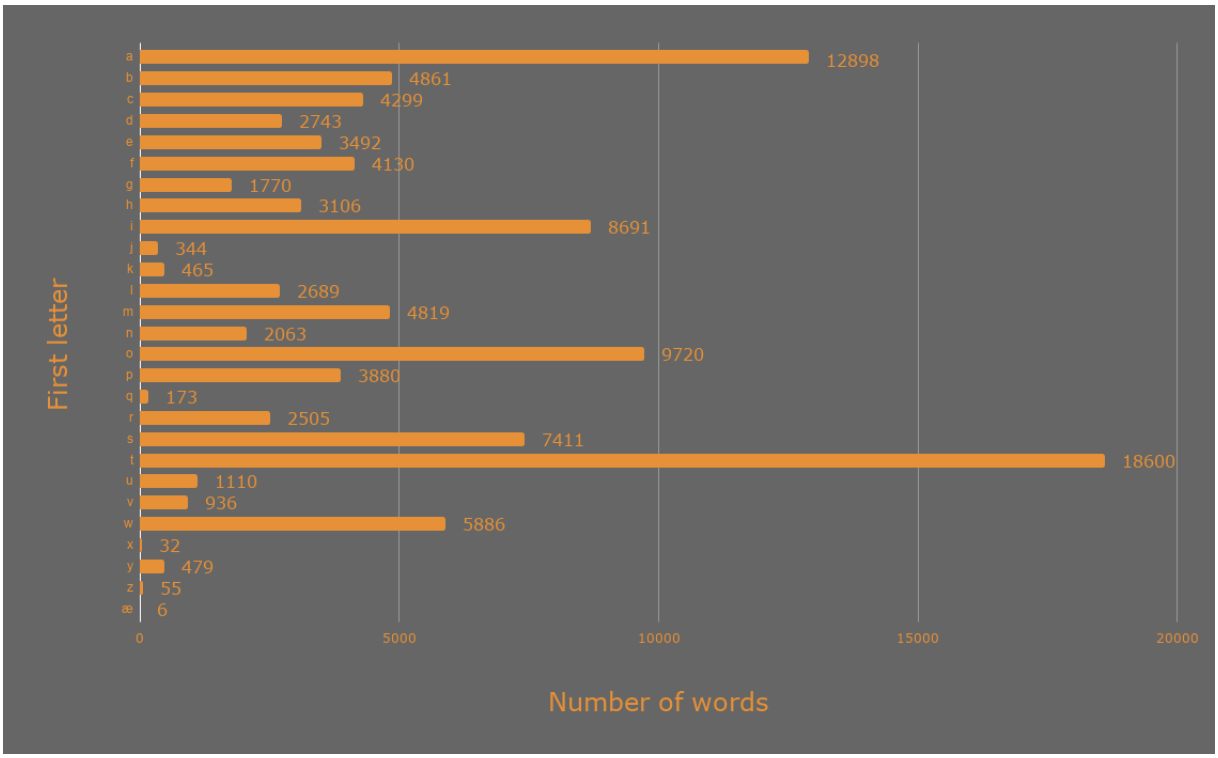
b. What is HDFS, and how is it different from the local filesystem on your virtual machine?

HDFS is the Hadoop Distributed File System which is a distributed scalable file system for the Hadoop framework. It has some properties that makes it very suitable for working with large data sets, properties that you would not find in your local file system. For example it is highly fault tolerant as data is splitted into smaller chunks and replicated across the multiple DataNodes. In a local filesystem your data is not replicated and instead stored in tree format. Another difference is that in order to access data in the HDFS you need to go through the NameNode to locate the data as HDFS works in a Master-Slave format between the NameNode and the DataNodes. Finally, HDFS supports steaming access to data.

Task 1.4

Make a plot that shows the counts for each letter and include that in your report.

In order to calculate and make the sought for plot the file WordCount.java was altered so that it instead counts the occurrences of words starting with the same first letter (non case-sensitive). By doing this the following plot was obtained:



Figur 1: *The occurrence of words starting with the same first letter in the textfile pg201417.txt (case-insensitive)*

The results above makes sense in terms of what one would expect from a large English text corpus. Less frequent letters like 'x', 'y','z' and 'q' having low counts while for example 't' has a very high count, likely due to words like 'the', 'this', 'that', 'there' etc.

Task 2.1

1a. Based on the twitter documentation in the above link, how would you classify the JSON-formatted tweets - structured, semi-structured or unstructured data?

I would classify the JSON-formatted tweet data as being semi-structured. It is not structured as the JSON format is too flexible to be arranged into tables. However, it is of course not unstructured data as all tweets consists of the same top level attributes (e.g. *id*, *text*, *created_at* and some possible nested child objects (e.g. *media*, *hashtags*)

1b. 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)?

One challenge could be that the traditional row-based RDBM would need to handle the nested fields that exists in the JSON tweet in our data set. This could complicate the queries necessary to do analysis. There are also attributes in our data set that can consist of arrays of values. For example the attribute *coordinates* which is formatted as geoJSON (longitude first, then latitude). This could also add complexity to the queries and make them less readable.

3. Plot a bar chart visualizing the counts of each pronoun, normalized by the total number of tweets

To count the number of tweets containing each pronoun the mapper file *mapper_twitter.py* was designed in such a way that it can handle JSON documents as the input from the STDIN, parse the incoming twitter JSON to a dictionary and check the occurrences of the pronouns. Any pronoun that existed in the twitter JSON was added to an additional dictionary whose content was sent to STDOUT.

In order to exclude retweets (a repost or forward of a message posted by another user) in this task an if-statement was included in the *mapper_twitter.py* file to check that the JSON tweets to include in the MapReduce did not contain the attribute *retweeted_status* which is, according to the twitter documentation, how to

to identify a retweet. [1].

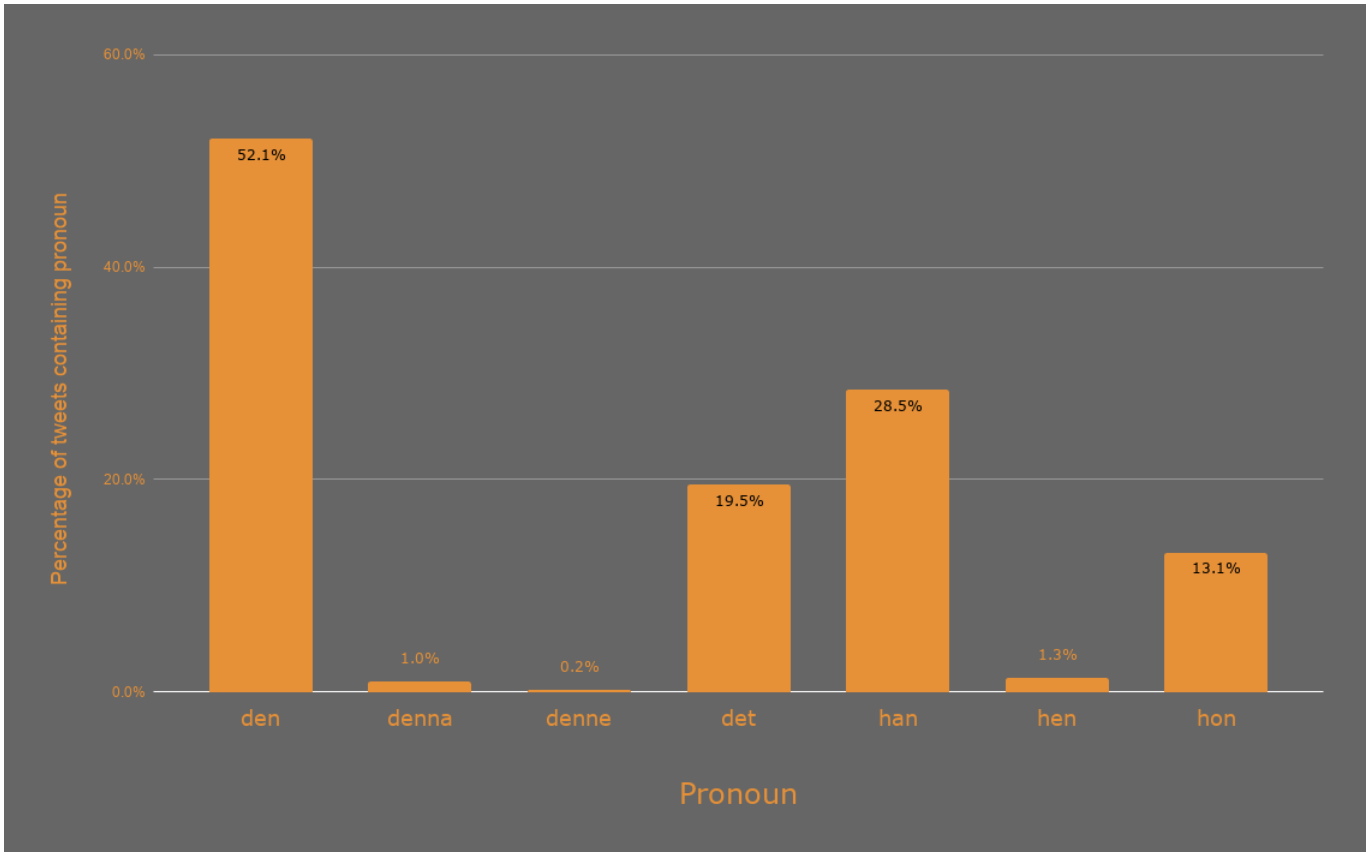
In order to normalize the counts of each pronoun the total number of non-retweets had to be identified. By counting the lines in all the tweet text files and given the fact that between each JSON was an empty line the total number of tweets was identified as being: 3 215 575. This was done with the following command in the command prompt:

```
$ wc -l <filename>
```

To identify the total number of retweets in the data set, the occurrences of the string 'retweeted_status' was counted in all the tweet data. Do note that this is not the optimal way as it is for example possible for this exact string to exist in the text key in a tweet. A more proper method could have been to simply alter the mapper file to count the number of retweets. This (naive) way of counting the number of retweets was done using the command:

```
$ tr ' ' '\n' < tweets_* | grep retweeted_status | wc -l.
```

Where tr replaces spaces with newlines, grep filters all resulting lines matching 'retweeted_status' and wc counts the remaining ones. 873 998 retweets were identified, meaning the total number of non-tweets to use for our normalization was: 2 341 577. This gives us the following plot:



Figur 2: *Chart displaying the percentage of tweets (excluding retweets) mentioning each of the pronouns (case-insensitive)*

The plot above shows the percentage of tweets in our data set that contained each of the listed pronouns. The results seem to be roughly in line with my perception of the frequency of pronouns in the Swedish language, with 'den', 'det', 'han' and 'hon' being the most common. Worth noting is that the masculine subjective singular case ('han') is more then twice as common to appear in a tweet as the feminine subjective singular case ('hon').

[1] https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/tweet

It is worth noting that there could potentially exist duplicate tweets in the data set which would affect the results. This was not taken into account in this analysis but could have been handled by utilizing the *id* attribute in the JSON tweets since this is a unique identifier for each tweet[2].

[2]https://developer.twitter.com/en/docs/twitter-api/v1/data-dictionary/object-model/tweet