Report of Assignment 2
Data Engineering

Lingkai Zhu

- Task 1.1: Word count example in local (standalone) mode

a. Look at the contents of the folder “output” - what are the files placed in

there? What do they mean?

Answer:

```
ubuntu@lingkaizhua2:~/wordcount/output$ ls
_SUCCESS  part-r-00000
```

_SUCCESS and part-r-00000 are placed in the output folder.

_SUCCESS means that the map is generated successfully, part-r-0000 stores the number of occurrence of each word given the input 20417.txt.utf-8.

b. How many times did the word ‘Discovery’ (case-sensitive) appear in the

text you analyzed?

Answer:

According to the file part-r-00000, it appears 5 times in the text I analyzed.

```
Dinosaur        1
Dinosaur,       1
Dinosaurs       3
Dinosaurs,      1
Diplodocus      1
Directly        1
Director        1
Disappearance   1
Discovery       5
Discs   1
Disguise        1
Dissipation     1
Distinctly      1
Distributed     2
Dividing        1
Do      3
Doctrine        1
Does    1
/Discovery
```

c. In this example we used Hadoop in “Local (Standalone) Mode”. What is

the difference between this mode and the Pseudo-distributed mode?

Answer:

The standalone mode is a non-distributed mode, where Hadoop is configured to run as a single Java process. On the contrary, the Pseudo-distributed mode is to be run on a single-node where each Hadoop daemon runs in a separate Java process. It is called Pseudo because it is not really distributed among several computers, and in this case, it stimulates the distribution in one single node.

- Task 1.2: Setup pseudo-distributed mode

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

Answer:

The core-site.xml file contains the configuration information such as the port number used for Hadoop instance, in this case, it is 9000, it can also contain the resource information such as the memory allocated for the file system, memory limit for storing the data, and the size of Read/Write buffers. Therefore, the role of core-site.xml here is to configure the core infrastructure property of the Hadoop instance.

The hdfs-site.xml file contains information such as the value of replication data, in this case, it is 1. It can also contain information such as namenode path and datanode path of the local file system. Therefore, the role of hdfs-site is to configure the property of the Hadoop Distributed File System (HDFS).

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

'jps'.

Answer:

```
ubuntu@lingkaizhu-a2:/usr/local/hadoop-3.3.1$ jps
4145 SecondaryNameNode
4419 Jps
3813 NameNode
3951 DataNode
```

NameNode is the centerpiece of the HDFS file system, keeps the directory tree of all files in the file system, and tracks where across the cluster the file data is kept. However, namenode does not store the data of the files itself.

SecondaryNameNode is a helper node for namenode, and the whole purpose of it is to have a checkpoint in HDFS. It creates checkpoints of the namespace by merging the edits file into the fsimage file. Also, SecondaryNameNode is not a redundancy of the NameNode.

DataNode is responsible for storing the actual data in the HDFS. When a DataNode starts up, it announces itself to the NameNode with the list of blocks it is responsible for and responds to requests from the NameNode for filesystem operations. Also, when a DataNode is down, it does not affect the availability of data because NameNode will arrange for replication of the blocks managed by the unavailable DataNode.

Jps lists all Java process on the machine, which is not specific to Hadoop.

- Task 1.3: Word count in pseudo-distributed mode

Result:

```
ubuntu@lingkaizhu-a2:~/wordcount$ vim WordCount.java
ubuntu@lingkaizhu-a2:~/wordcount$ /usr/local/hadoop-3.3.1/bin/hdfs dfs -tail output1/part-r-00000
ww.gutenberg.org        2
yacht,  1
yard    1
yards   3
year    16
year!   1
year,   5
year.   4
year.]  1
years   73
years,  14
years--but      1
years.  15
years;  2
yellow  2
yellow, 4
```

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

Answer:

The class TokenizerMapper extends the class Mapper which is used to Maps input key/value pairs to a set of intermediate key/value pairs. The TokenizerMapper will process the input key/value pairs, in this case, key is the position of the phrases in the file and value is the phrases that we are interested. StringTokenizer will break a string into tokens, namely break the phrases into separate words, finally store the result as the form (word, number of occurrence) to context. For example, if input is (0, “I am”), we will get the output (I, 1) and (am, 1)

The class IntSumReducer extends the class Reducer which reduces a set of intermediate values which share a key to a smaller set of values. The IntSumReducers will combine multiple map outputs (word, number of occurrence) with the same word into one map output. For example, if the map outputs are (happy, 1) and (happy, 1), then IntSumReducers will combine them and output (happy, 2)

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

virtual machine?

Answer:

HDFS (Hadoop Distributed File System) is a distributed file system that handles large data sets running on commodity hardware. It is used to scale s single Apache Hadoop cluster to hundreds of nodes. Compared with the local filesystem, HDFS has the following advantages:

1. Fast recovery from hard failures
   As it is shown in the Pseudo-distributed mode, there are replication data on different DataNode. Therefore, failure of at least one server can be detect, and data can be recovered quickly.
2. Access to streaming data
   HDFS is designed to handle big data throughput rates.
3. Accommodation of large data sets
   HDFS accommodates application that have data sets typically gigabytes to terabytes in size, which are much larger than the local filesystem. HDFS provides high aggregate data bandwidth for the purpose of big data handing.
4. Portability
   HDFS is designed to be portable across multiple hardware and to compatible with various operating systems.

- Task 1.4: Modified word count example

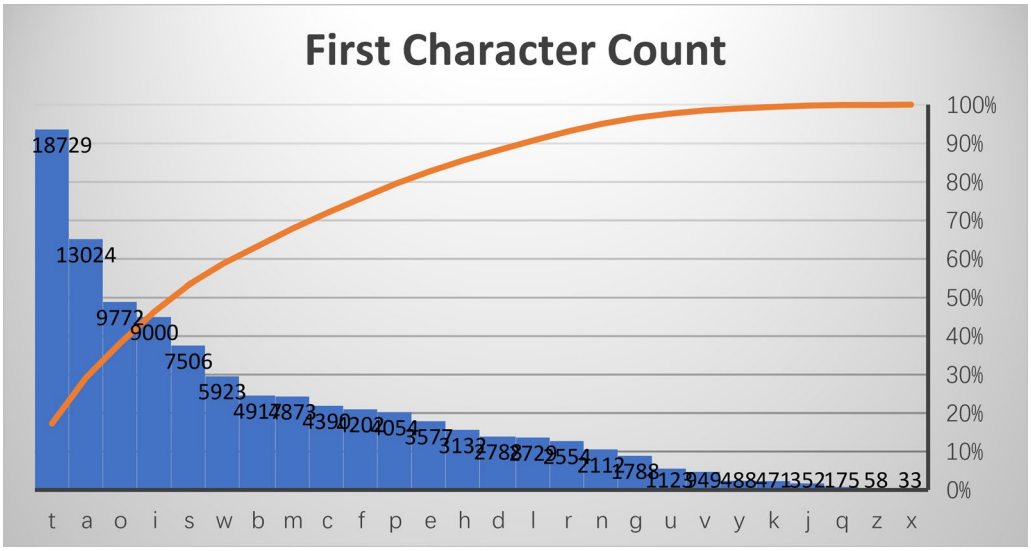
1. Modify the above example code in WordCount.java so that it, based on the same input files, counts the occurrences of words starting with the same first letter (non case-sensitive). The output of the job should thus be a file containing lines like:

a number_of_words_starting_with_a_or_A

Answer:

My idea is to change the original word-value pairs first-character-value pairs, when a word is identified instead of passing the word as output (word, value), I take the first character as the output (first-character, value). Before taking the word, I use the regular expression to derive the word from the string containing punctuations, for example, to get the word ‘book’ from ‘[book]’.

2. Make a plot that shows the counts for each letter and include that in your report. Do not include the entire output file.



The words starting character t are the most.

- Task 1.5: NoSQL and MongoDB

1. One example of JSON formatted data is Twitter tweets: https://dev.twitter.com/overview/api/tweets. Based on the twitter documentation, how would you classify the JSON-formatted tweets - structured, semi-structured or unstructured data?

Answer:

Based on the twitter documentation, I would describe the JSON-formatted tweets as semi-structured data because we have a mixture of structured data and unstructured data to observe. The text, video and images are unstructured data which are stored in data

lakes or NOSQL, while the time a tweet was made, id, id_str and user are structured data each of which can be easily mapped into a column in a table.

2. Elaborate on pros and cons for SQL and NoSQL solutions, respectively. Give some examples of particular data sets/scenarios that might be suitable for these types of databases. (expected answer length: 0.5 A4 pages)

Answer:

SQL solutions are mainly coming under Relational Database, and need vertical scaling. SQL solutions are table based and can be accessed by the SQL language. SQL is used to store structured data.

Pros:

- Flexible queries: Enable support for diverse workloads
- Reduced data storage footprint: a reduced footprint maximizes database performance and resource usage.
- Strong and well-understood data integrity semantics

Cons:

- Rigid data models: requires a predefined schema for unstructured data before using SQL to manipulate data
- Limited horizontal scalability
- Single point of failure
- Costly vertical scaling: vertical scaling requires to add more hardware resources, such as GPU an memory to the existing machine.

NoSQL databases mostly come under non-relational or distributed database, and are horizontal scalable. NoSQL is based upon document approach such as key-value pairs and the access to NoSQL is much more dynamic than SQL. NoSQL is used to store unstructured JSON files.

Pros:

- Cheap and Quick Scalable and highly available: support seamless, online horizontal scalability without significant single points of failure. And in horizontal scaling, because each node contains only part of the data, allowing to add more machines to the existing distributed system.
- Flexible data models
- Dynamic schema for unstructured data: allows storing data before applying schema
- High performance
- High-level data abstraction

Cons:

- Queries of NoSQL are not as powerful as compared to SQL query language
- Lack true ACID transactions
- May exhibit lost writes and other form of data loss

Examples:

When the data is relational and the relationship between data sets is well-defined and highly navigable, it might be more suitable to use SQL solutions.

When working with complex queries and reports in an intensive environment, SQL is better.

When working with large amount of unstructured data, it might be more suitable to use NoSQL solutions.

When the data is needed to store in real time, NoSQL is better because it does not require schemas.

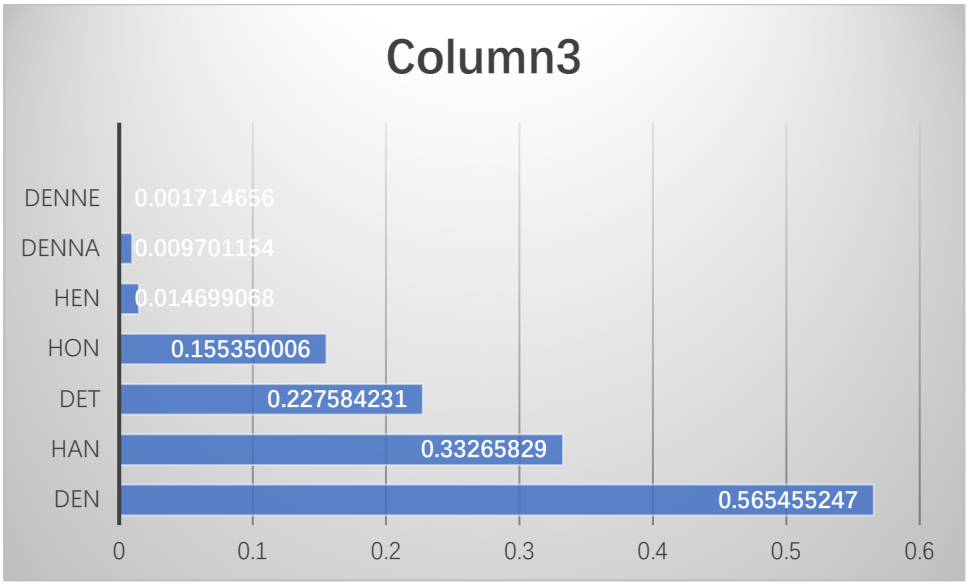
- Task 2.1: Analyzing twitter data using Hadoop streaming and Python (awards up to 1 point)

Implementation Description:

First of all, because the machine is using Python3 now, I have to change the *#!/usr/bin/env python* in the tutorial to *#!/usr/bin/env python3*. In the mappy.py, because the json object in the txt file is separated by an empty, I get the json object by dropping empty lines, then I get the 'text' attribute and use regular expression to get the wanted pronouns from lowercase text.

After mapping the pronouns in mappy.py, the resulting key-value pairs will be fed into the reducer.py and get combined and reduced.

Result:



Total number of unique tweets: 2341577.

- Task 2.2: NoSQL and MongoDB

1. Redo the analysis from Task 2.1, now using MongoDB.

- Use the exact same data as in Task 2.1
- Some suggested options include using the Mongo Shell or PyMongo
- This is an open problem, it is up to you to experiment with MongoDB and develop a solution to the problem that makes efficient use of MongoDBs capabilities. Add complexity to the problem if you desire. Note that there is no requirement to use a specific language or interface - the implementation is up to you.
- Present plots corresponding to those you obtained for Task 2.1

2. Answer the following question:

Motivate your chosen implementation and how you did it. What are some pros and cons of the MongoDB solution compared to the implementation you did in Task 2.1?

Answer:

First, insert the tweet data into MongoDB, then I use the PyMongo's insert_many methond to insert the data into MongoDB. To prevent MongoDB from crash and to simplify the data structure, here I only consider insert the text of tweets without retweeted_status.

Second, after successfully inserting the data into MongoDB, my initial thought is to implement the map reduce procedures similar to what I did with Hadoop in Task 2.1, but after reading the MongoDB documents, it turned out the map-reduce option is deprecated in MongoDB 5.0, and now there is an aggregation pipeline which provides better performance and usability than a map-reduce operation. Therefore, I decide to imbed regular expression in the MongoDB aggregation pipeline operations. The shell commands are included in 'mongoDBPronouncountShell.txt'.

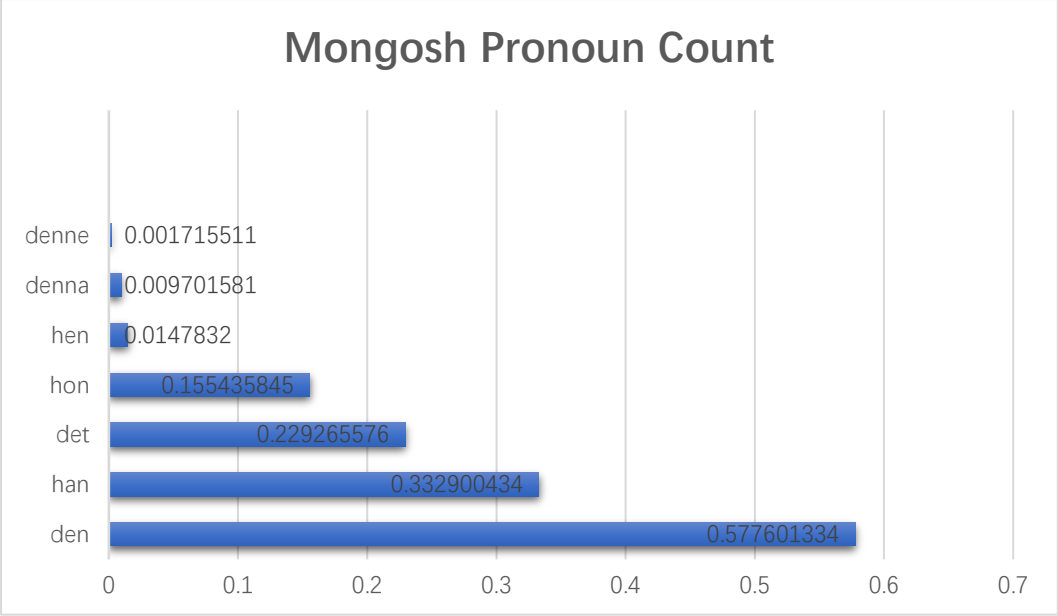
It is also possible to execute the command in Pymongo.

Result:

```
[
  { _id: 'den', totalOccurences: 1352498 },
  { _id: 'han', totalOccurences: 779512 },
  { _id: 'det', totalOccurences: 536843 },
  { _id: 'hon', totalOccurences: 363965 },
  { _id: 'hen', totalOccurences: 34616 },
  { _id: 'denna', totalOccurences: 22717 },
  { _id: 'denne', totalOccurences: 4017 }
]
```

Total number of unique tweets: 2341577.

Plot:



The result is basically the same to what I did in task 2.1 with Hadoop.

Pros:

1. MongoDB executes pronoun count procedure faster than Hadoop as MongoDB is written in C++.
2. MongoDB now offers more powerful and efficient aggregation method which is better than the original map reduce method.

Cons:

1. It is time-consuming to insert the data into MongoDB especially when you have a very large data set and lack of enough memory to handle it.
2. MongoDB requires lots of memory as all files have to be mapped from disk to memory, therefore in storing large size of tweets data, I have to open the allowDiskUse() feature to use temporary files on disk to store data exceeding the 100 megabyte system memory limit.