COMP4434 Homework 1

ZHANG Caiqi 18085481d

February 8, 2021

1 Question 1

- Volume: We can firstly find that the amount of data is large from the text. It says that the peak volume of Sina Weibo posts reached a new high in 2020. There are over thirty thousands of Weibo been posted in the midnight of Chinese New Year 2020.
- Velocity: From the text, we know that the total number of social Internet users in China will increase by 4.8% in 2020, reaching 859.1 million. Therefore, we may conclude that the data is generated in high velocity because a large number of people will generate a large amount of data everyday.
- Variety: The text also implicitly indicate there are many kinds of data been generated everyday. This is because that the Weibo posts consists of many kinds of information, such as text, video, picture and audio.

2 Question 2

2.1 a)

We know the page rank equation is as follows with $d = 0.5$.

$$PR(A) = \underbrace{(1-d)}_{\text{contributed by ``teleport'' operation}} + d\left(\frac{PR(T_1)}{C(T_1)} + \cdots + \frac{PR(T_n)}{C(T_n)}\right)$$

Therefore,
PR(A) = 0.5 + 0.5(PR(C) + PR(B)/2)
PR(B) = 0.5 + 0.5(PR(A)/2)
PR(C) = 0.5 + 0.5(PR(A)/2 + PR(B)/2)
After calculation, we can get the following table:

Iteration	PR(A)	PR(B)	PR(C)
0	1	1	1
1	1.25	0.75	1
2	1.188	0.813	1
3	1.203	0.797	1

2.2 b)

It takes around 26 iterations to converge.
The final page rank values for A, B, and C are 1.2, 0.8, 1.

2.3 c)

From the pseudocode, the Map() function will calculate the intermediate result by $emit(page, PR(key)/sizeof(Adj[key])$.
PR(A) = PR(B) = PR(C) = 1, Adj[A] = Adj[B] = 2, Adj[C] = 1. Therefore, using the above formula, the intermediate result is as follows:

B	0.5
C	0.5
A	0.5
C	0.5
A	1

3 Question 3

3.1 a)

The Pseudo codes are in Figure 1.
We are going to solve this problem by Map-Reduce using the following method: In the Map() part, for every pair of (Child, Parent), we generate the intermediate results by connecting their names and relationship For example, for pair (Tom, Lucy), we will also generate an intermediate result (Tom-Lucy, child-parent). Moreover, for the same pair, we will generate a reverse one, which is (Lucy-Tom, parent-child) in this example. Each mapper will work on half of the census data, therefore, we can get the intermediate results in the blue block in Figure 2.
After sorting the records, we will go to the Reduce() part. As we have two reducers, we can let the Reducer 1 to deal with items with first letter A-L and the other one for M-Z. The algorithm is to check whether a family member have both parent and child. For example, let's see the records starting with "Jack", which are (Jack-Alice, child_parent), (Jack-Jesse, child_parent)), (Jack-Jone, parent_child), (Jack-Tom, parent_child). Because that Jack has both parents and children, his parents and children can form many (grandchild, grandparent) pairs. The output of each reducer is also shown in the orange block in Figure 2.

```
Map(key, value) {

    // key: table name
    // value: the content of the table (census data)

    for each row in value {
        //Record the items in child-parent order:
        emit(memberA+memberB, "child-parent")
        //Record the items in parent-child order:
        emit(memberB+memberA, "parent-child")
    }
}

Reduce(key, values) {

    // key: family member pairs connecting with "-" (memberA-memberB)
    // values: the relationship between the two members

    result_dict = {}

    for every memberA:
        initialize parent_list = [], child_list = []

        if relationship == "parent_child":
            child_list.append(memberB)
        if relationship == "child_parent":
            parent_list.append(memberB)

        result_dict[memberA] = [child_list, parent_list]

    for every key in result_dict:
        emit(grandchild, grandparent)

}
```

Figure 1: Pseudo codes

3.2 b)

Figure 3 is the screenshot of the results. The codes are run in Mac OS, Pycharm.

Figure 2: Intermediate results

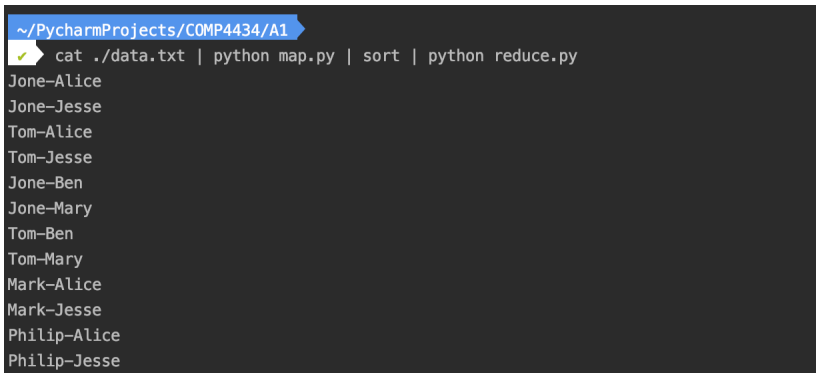

Figure 3: Result screenshot