

Assignment 4

Problem-2

Task 1 : Data Processing using Spark

Mapreduce Algorithm:

Step 1: Connect to the instance created on GCP during task 1.

Step 2: Push all the text files to the gitlab repository and clone this repository in the GCP instance to retrieve all the text files containing the Twitter data.

Step 3: open spark shell from this directory and create RDD containing all the files with the command : `val rdd = spark.sparkContext.textFile("*").`

Step 4: Split the data by space to fetch all the words.

`val splitdata = data.flatMap(line => line.split(" "));`

Step 5: Perform the map operation. `val mapdata = splitdata.map(word => (word,1));`

Step 6: Perform the reduce operation. `val reducedata = mapdata.reduceByKey(_+_);`

Step 7: Create a list of words that we need the word count for : `val keyword_list :`

`List[String] = List("flu", "snow", "cold").`

Step 8: Fetch the word count by iterating in a for each loop and matching the data with the list of words created in Step 7.

Word count:

Snow: 318

Flu: 87

Cold: 831

Screenshot of the word count:

```
Using Scala version 2.12.10 (OpenJDK 64-Bit Server VM, Java 11.0.14)
Type in expressions to have them evaluated.
Type :help for more information.

scala> reducedata.foreach( case (keyword, freq) => { if(keyword_list.contains(keyword)) {println(s"$keyword happens $freq")}})
<console>:24: not found: value reducedata
reducedata.foreach( case (keyword, freq) => { if(keyword_list.contains(keyword)) {println(s"$keyword happens $freq")}})
^
scala>:24: error: not found: value keyword_list
reducedata.foreach( case (keyword, freq) => { if(keyword_list.contains(keyword)) {println(s"$keyword happens $freq")}})
^

scala> val rdd = spark.sparkContext.textFile("")
rdd: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[1] at textFile at <console>:23

scala> rdd.collect;
res1: Array[String] = Array(["data":[{"metadata":{"result_type":"recent","iso_language_code":"en"},"in_reply_to_status_id_str":null,"in_reply_to_status_id":null,"created_at":"Sun Mar 13 13:53:34 +0000 2022","in_reply_to_user_id_str":null,"source":"Ca href=\"https://d1vr.it.com/V\" rel=\"nofollow\">d1vr.it</a>","retweeted_status":{"extended_entities":{"media":[{"display_url":"pic.twitter.com/gqsh09bh5","indices":{"lower":106,"upper":129},"size":{"small":{"w":500,"h":500,"resize":"fit"},"large":{"w":500,"h":500,"resize":"fit"},"thumb":{"w":150,"h":150,"resize":"crop"},"medium":{"w":500,"h":500,"resize":"fit"},"id_str":"1503005592138358786","expanded_url":"https://twitter.com/iCoolCreate/status/1503005593887334404/photo/1","media_url_https":"https://pbs.twimg.com/media/FNu_m27V...

scala> val splitdata = rdd.flatMap(line => line.split(" "));
splitdata: org.apache.spark.rdd.RDD[String] = MapPartitionsRDD[2] at flatMap at <console>:25

scala> splitdata.collect;
res2: Array[String] = Array(["data":[{"metadata":{"result_type":"recent","iso_language_code":"en"},"in_reply_to_status_id_str":null,"in_reply_to_status_id":null,"created_at":"Sun, Mar, 13, 13:53:34, +0000, 2022","in_reply_to_user_id_str":null,"source":"Ca href=\"https://d1vr.it.com/V\" rel=\"nofollow\">d1vr.it</a>","retweeted_status":{"extended_entities":{"media":[{"display_url":"pic.twitter.com/gqsh09bh5","indices":{"lower":106,"upper":129},"size":{"small":{"w":500,"h":500,"resize":"fit"},"large":{"w":500,"h":500,"resize":"fit"},"thumb":{"w":150,"h":150,"resize":"crop"},"medium":{"w":500,"h":500,"resize":"fit"},"id_str":"1503005592138358786","expanded_url":"https://twitter.com/iCoolCreate/status/1503005593887334404/photo/1","media_url_https":"https://pbs.twimg.com/media/F...

scala> val mapdata = splitdata.map(word => (word,1));
mapdata: org.apache.spark.rdd.RDD[(String, Int)] = MapPartitionsRDD[3] at map at <console>:25

scala> mapdata.collect;
res3: Array[(String, Int)] = Array(["data":[{"metadata":{"result_type":"recent","iso_language_code":"en"},"in_reply_to_status_id_str":null,"in_reply_to_status_id":null,"created_at":"Sun, 13, (Mar, 13, 13:53:34, 1), (+0000, 1), (2022,\"in_reply_to_user_id_str\":null,\"source\":\"Ca, 1), (href=\"https://d1vr.it.com/V, 1), (rel=\"nofollow\">d1vr.it</a>\", \"retweeted_status\": {\"extended_entities\": {\"media\": [{\"display_url\": \"pic.twitter.com/gqsh09bh5\", \"indices\": {\"lower\": 106, \"upper\": 129}, \"size\": {\"small\": {\"w\": 500, \"h\": 500, \"resize\": \"fit\"}, \"large\": {\"w\": 500, \"h\": 500, \"resize\": \"fit\"}, \"thumb\": {\"w\": 150, \"h\": 150, \"resize\": \"crop\"}, \"medium\": {\"w\": 500, \"h\": 500, \"resize\": \"fit\"}, \"id_str\": \"1503005592138358786\", \"expanded_url\": \"https://twitter.com/iCoolCreate/status/1503005593887334404/photo/1\", \"media_url_https\": \"https://pbs.twimg.com/media/F...

scala> val reducedata = mapdata.reduceByKey(_+_);
reducedata: org.apache.spark.rdd.RDD[(String, Int)] = ShuffledRDD[4] at reduceByKey at <console>:25

scala> reducedata.collect;
res4: Array[(String, Int)] = Array(["https://t.co/7jPz5f3hop","place":null,"lang":"en","favorite":false,"possibly_sensitive":false,"coordinates":null,"truncated":true,"entities":{"urls":[{"display_url":"twitter.com/i/web/status/1u2026","indices":{"lower":116,"upper":139},"expanded_url":"https://twitter.com/i/web/status/15029828401067777","url":"https://t.co/7jPz5f3hop"}],"hashtags":[],"symbols":[],"contributors":null,"user":{"utc_offset":null,"friends_count":148,"profile_image_url_https":"https://pbs.twimg.com/profile_images/150192722529212416/54rxm3Kl_normal.jpg","listed_count":2015,"profile_background_image_url":null,"default_profile_image":false,"favourites_count":10202,"description":"xx, 7), (addict., 134), (@Nannook2021, 14), (thick, 6), (spicy, 21), ...

scala> val keyword_list = List[String] = List("flu", "snow", "cold")
keyword_list: List[String] = List(flu, snow, cold)

scala> reducedata.foreach( case (keyword, freq) => { if(keyword_list.contains(keyword)) {println(s"$keyword happens $freq")}})
snow happens 318
flu happens 87
cold happens 831=====
(26 + 2) / 33)

res4: Array[(String, Int)] = Array(["https://t.co/7jPz5f3hop","place":null,"lang":"en","favorite":false,"possibly_sensitive":false,"coordinates":null,"truncated":true,"entities":{"urls":[{"display_url":"twitter.com/i/web/status/1u2026","indices":{"lower":116,"upper":139},"expanded_url":"https://twitter.com/i/web/status/15029828401067777","url":"https://t.co/7jPz5f3hop"}],"hashtags":[],"symbols":[],"contributors":null,"user":{"utc_offset":null,"friends_count":148,"profile_image_url_https":"https://pbs.twimg.com/profile_images/150192722529212416/54rxm3Kl_normal.jpg","listed_count":2015,"profile_background_image_url":null,"default_profile_image":false,"favourites_count":10202,"description":"xx, 7), (addict., 134), (@Nannook2021, 14), (thick, 6), (spicy, 21), ...

scala> val keyword_list = List[String] = List("flu", "snow", "cold")
keyword_list: List[String] = List(flu, snow, cold)

scala> reducedata.foreach( case (keyword, freq) => { if(keyword_list.contains(keyword)) {println(s"$keyword happens $freq")}})
snow happens 318
flu happens 87
cold happens 831=====
(26 + 2) / 33)
```

Task 2 : Data Visualization using Graph Database

Cypher queries:

//Creating Flu nodes

```
CREATE (n:FluNode{ name:'flu',types:'Influenza A,Influenza B,Influenza C'})
CREATE (n:flu{ name:'Influenza A', fluProperty:'antigenic',occurrenceInTweets:'13',
types:'hemagglutinin,neuraminidase', symptoms: 'cough,runny nose,fever,fatigue'})
CREATE (n:flu{ name:'Influenza B', fluProperty:'antigenic',occurrenceInTweets:'25',
types:'types: Victoria,Yamagata', symptoms:'fever,chills,sore throat,cough'})
```

```
CREATE (n:flu{name:'Influenza C', fluProperty:'genetic and  
antigenic',occurenceInTweets:'29', types:'hemagglutinin,neuraminidase', symptoms:'dry  
cough, muscle pain,achiness'})
```

//Creating Snow nodes

```
CREATE (n:SnowNode{name:'snow',types:'Blizzard,Snow storm,Snow Burst'})  
CREATE (n:snow{typeOfSnowFall:'Blizzard', name:'Snowflakes', location:'Russia and  
central and northeastern Asia, northern Europe, Canada, the northern United States, and  
Antarctica', inches:'36'})  
CREATE (n:snow{typeOfSnowFall:'Snow storm', name:'Polycrystals',  
location:'Northeastern United States, Eastern United States, Tibet, Eastern Canada',  
inches:'39'})  
CREATE (n:snow{typeOfSnowFall:'Snow Burst', name:'Graupel', location:'United  
States, Europe and Canada', inches:'31'})
```

//Creating Cold nodes

```
CREATE (n:ColdNode{name:'cold',types:'Freezing Weather,Diseases'})  
CREATE (n:cold{type:'freezing weather',name:'Freezing Weather',location:'Eastern  
Antarctic Plateau,Russia Greenland,Coastal regions',windSpeed:'>3mph',season:'winter'})  
CREATE (n:cold{type:'diseases',name:'Pneumonia',symptoms:'Chest  
pain,Cough,Fever,Shortness of breath',types:'Bacterial,Fungal,Viral'})  
CREATE (n:cold{type:'diseases',name:'Sinusitis',symptoms:'nasal  
inflammation,swelling,pain,blocked nose',types:'Accute,Subaccute,Chronic'})
```

//Generating relationships with cold nodes

```
MATCH (a:ColdNode), (b:cold) WHERE a.name="cold" AND b.name="Freezing  
Weather" CREATE (b)-[r:CONTEXT_OF]->(a)  
MATCH (a:ColdNode), (b:cold) WHERE a.name="cold" AND b.name="Pneumonia"  
CREATE (b)-[r:TYPE_OF]->(a)  
MATCH (a:ColdNode), (b:cold) WHERE a.name="cold" AND b.name="Sinusitis"  
CREATE (b)-[r:TYPE_OF]->(a)
```

//Generating relationships with snow nodes

```
MATCH (a:SnowNode), (b:snow) WHERE a.name="snow" AND  
b.typeOfSnowFall="Blizzard" CREATE (b)-[r:TYPE_OF]->(a)  
MATCH (a:SnowNode), (b:snow) WHERE a.name="snow" AND  
b.typeOfSnowFall="Snow storm" CREATE (b)-[r:TYPE_OF]->(a)
```

```
MATCH (a:SnowNode), (b:snow) WHERE a.name="snow" AND  
b.typeOfSnowFall="Snow Burst" CREATE (b)-[r:TYPE_OF]->(a)
```

//Generating relationships with flu nodes

```
MATCH (a:FluNode), (b:flu) WHERE a.name="flu" AND b.name="Influenza A"  
CREATE (b)-[r:TYPE_OF]->(a)
```

```
MATCH (a:FluNode), (b:flu) WHERE a.name="flu" AND b.name="Influenza B"  
CREATE (b)-[r:TYPE_OF]->(a)
```

```
MATCH (a:FluNode), (b:flu) WHERE a.name="flu" AND b.name="Influenza C"  
CREATE (b)-[r:TYPE_OF]->(a)
```

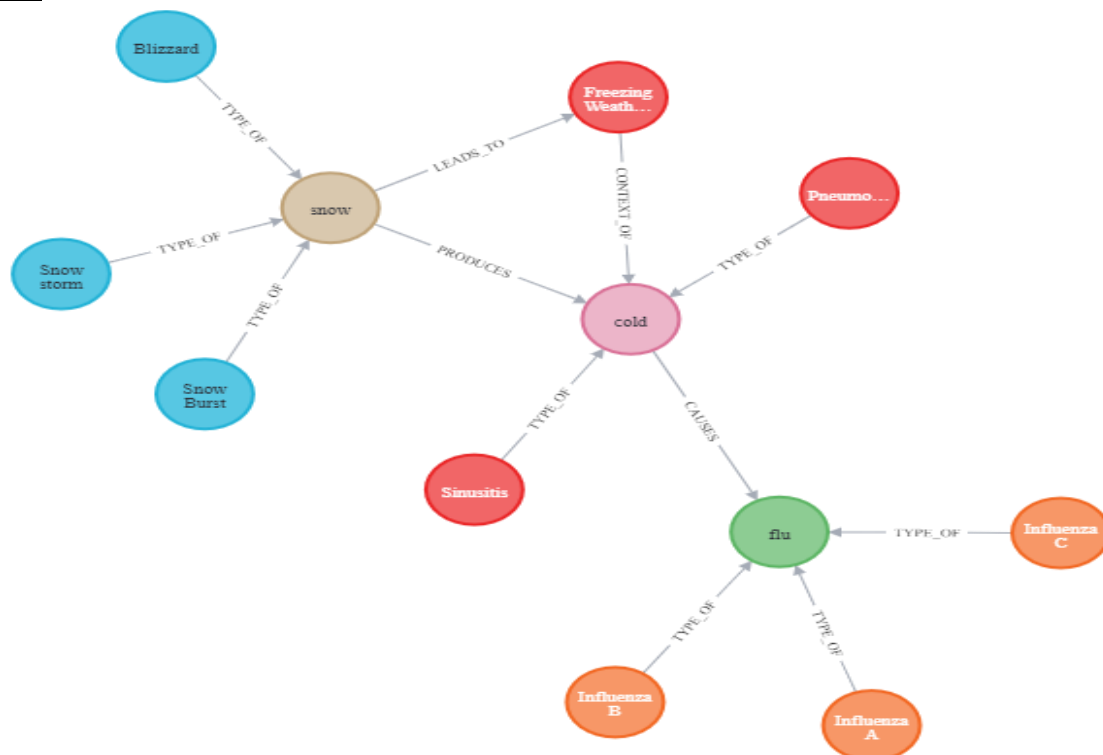
//Generating overall relationships

```
MATCH (a:SnowNode), (b:ColdNode) WHERE a.name="snow" AND b.name="cold"  
CREATE (a)-[r:PRODUCES]->(b)
```

```
MATCH (a:FluNode), (b:ColdNode) WHERE a.name="flu" AND b.name="cold"  
CREATE (b)-[r:CAUSES]->(a)
```

```
MATCH (a:SnowNode), (b:cold) WHERE a.name="snow" AND b.name="Freezing  
Weather" CREATE (a)-[r:LEADS_TO]->(b)
```

Graph:



References:

- [1] "Apache Spark Word Count Example - Javatpoint", *www.javatpoint.com*, 2022. [Online]. Available: <https://www.javatpoint.com/apache-spark-word-count-example>. [Accessed: 13-Mar- 2022].