Global Clustering Coefficient

Συστήματα Διαχείρισης Μεγάλης Κλίμακας

Χαροκόπειο Πανεπιστήμιο

Τμήμα Πληροφορικής και Τηλεματικής

Καραγιαννόπουλος Αριστείδης *A.M 17402*

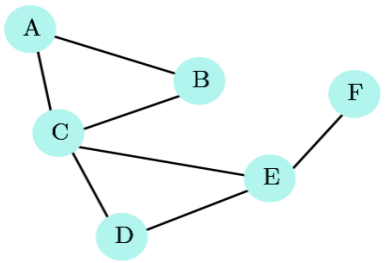
Κατσιάνος Δημήτριος *A.M 17403*

ΑΘΗΝΑ,
ΙΟΥΝΙΟΣ 2019

Περιεχόμενα

1. Εισαγωγή

Στην παρούσα αναφορά εξετάζουμε πως υπολογίσαμε με το σύστημα Spark τη μετρική clustering coefficient (συντελεστής ομαδοποίησης) για τους κόμβους ενός δικτύου. Τα δίκτυα τα οποία έχουν επιλεχτεί για μελέτη είναι “Pennsylvania road network” το οποίο είναι διαθέσιμο στη διεύθυνση “https://snap.stanford.edu/data/roadNet-PA.html” και “Social circles: Facebook” το οποίο είναι διαθέσιμο στην διεύθυνση “https://snap.stanford.edu/data/ego-Facebook.html”. Για λόγους testing αρχικά χρησιμοποιήσαμε δοκιμαστικό dataset το οποίο στηρίχτηκε στο παρακάτω δίκτυο (Εικόνα 1) όπου A =1, B =2, C=3, D=4, E=5, F=6. (αρχείο example_fof). Ολόκληρο το project μαζί με τα δεδομένα εισόδου, εξόδου τον κώδικα και οδηγίες για την υλοποίηση του υπάρχουν και στη διεύθυνση “ https://gitlab.com/bigdataspartkhua/clustering-coefficient-spark-hua.git”.



Εικόνα 1

2. Επεξήγηση Κώδικα

Αρχικά δημιουργούμε ένα maven project. Το πακέτο του project μας είναι το org.spark.clustering.coefficient και η κλάση μας είναι η ClusCoe. Στη συνέχεια δημιουργούμε ένα Pattern με ονομασία Space έτσι ώστε να πούμε στο πρόγραμμά μας πως θα κάνει split τα δεδομένα από το αρχείο. Επίσης, χρειάστηκε στο dataset “Pennsylvania road network” να επεξεργαστούμε το δεδομένα, διότι ο διαχωρισμός γινόταν με tab και όχι με κενό. Δημιουργούμε την κλάση μας με ονομασία “ClusCoe”.

```
public class ClusCoe {
   private static final Pattern SPACE = Pattern.compile(" ");

   public static JavaPairRDD<String, Double> compute(JavaRDD<String> lines) {
```

2.1.Δημιουργία πρώτου Rdd (nodePairRddFirst)

Αρχικά ξεκινάμε δημιουργώντας το πρώτο RDD με ονομασία “nodePairRddFirst” στο οποίο γράφουμε τα δεδομένα μας από το αρχείο που έχουμε ανεβάσει στον server.

```
JavaPairRDD<Long, Long> nodePairRddFirst = lines.flatMapToPair((String s) -> {
```

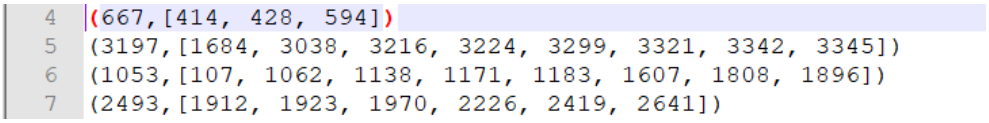
```
    String x[] = SPACE.split(s);
    long source = Long.parseLong(x[0]);
    long target = Long.parseLong(x[1]);

    return Arrays.asList(new Tuple2<>(source, target), new Tuple2<>(target, source)).iterator();
});
nodePairRddFirst.saveAsTextFile("/home/vagrant/nodePairRddFirst");
nodePairRddFirst.cache();
```

Το παραγόμενο Rdd το κάνουμε cache και το αποθηκεύουμε στον αντίστοιχο φάκελο. Το παραγόμενο Rdd είναι της μορφής :

Εικόνα 2

Έπειτα δημιουργούμε ένα Rdd το οποίο θα περιέχει τους γειτονικούς κόμβους για κάθε κόμβο και θα είναι της μορφής <node,[node1,node2….]>

```
JavaPairRDD<Long, Iterable<Long>> adjacencyList = nodePairRddFirst.groupByKey();
```



Εικόνα 3

2.2.Δημιουργία Rdd(pCombinationPair)

Ακολούθως, θα δημιουργήσουμε ένα Rdd το οποίο θα περιέχει όλους τους δυνατούς συνδυασμούς ζευγαριών των γειτόνων για κάθε κόμβο. Το Rdd θα είναι της μορφής <[node1,node2],node>. Το πρώτο if (nodeF != nodeS) έχει μπει ώστε να αποφύγουμε να βάλει στο Rdd ζευγάρια με ίδιους κόμβος π.χ <[1,1],2>. Το δεύτερο if (nodeS >nodeF) μας βοηθάει ώστε να αποφύγουμε να δημιουργήσουμε όμοια ζευγάρια αντίστροφα π.χ. <[1,2],1> και <[2,1],1>

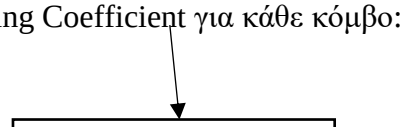
```
JavaPairRDD<Tuple2<Long, Long>, Long> pCombinationPair = adjacencyList.flatMapToPair(s -> {

    List<Tuple2<Tuple2<Long, Long>, Long>> listCombinationPair = new ArrayList<>();

    Long nodeM = s._1;
    for (long nodeF : s._2()) {
        for (long nodeS : s._2()) {

            if (nodeF != nodeS) {
                if (nodeS > nodeF) {
                    listCombinationPair.add(new Tuple2<>(new Tuple2<>(nodeF, nodeS), nodeM));
                }
            }
        }
    }
```

```
    return listCombinationPair.iterator();
});
```

```
((3437,3463),3558)
((3437,3493),3558)
((3437,3506),3558)
((3437,3568),3558)
((3437,3584),3558)
((3437,3593),3558)
((3437,3620),3558)
((3437,3643),3558)
((3437,3646),3558)
((3437,3648),3558)
```

Εικόνα 4

Σε αυτό το σημείο έχουμε φτιάξει 2 Rdds τα οποία θα μας βοηθήσουν να βρούμε το κλάσμα του τύπου Clustering Coefficient για κάθε κόμβο:



$$\frac{1+1+\frac{1}{3}+1+\frac{1}{3}+0}{} = \frac{11}{18}$$

Εικόνα 5

2.3.*Δημιουργία Rdd για την εύρεση του παρονομαστή (calcuNodeDegreeDen)*

Το ένα Rdd θα μας βρει τον παρονομαστή του κλάσματος και θα είναι της μορφής <node,sum>:

```
JavaPairRDD<Long, Double> calcuNodeDegreeDen = nodePairRddFirst.mapToPair(s -> {

    return new Tuple2<>(s._2, 1);

}).reduceByKey((a, b) -> {
    return a + b;

}).mapToPair((s) -> {

    Double s2 = new Double(s._2);
    Long s1 = s._1;

    Double calcDegree = (s2 * (s2 - 1)) / 2;

    return new Tuple2<>(s1, calcDegree);
});
```

```
(3586,946.0)
(1084,105.0)
(3558,703.0)
(1410,210.0)
(3456,1275.0)
(3702,703.0)
(3272,946.0)
(3066,91.0)
(4038,36.0)
(1894,1953.0)
```

Εικόνα 6

2.4. Δημιουργία Rdd για την εύρεση του αριθμητή (calcuNodeDegreeDen)

Το άλλο Rdd θα μας βρει τον αριθμητή του κλάσματος και θα είναι της μορφής <node,sum>. Για να μπορέσουμε να φτάσουμε στο Rdd "nodeNeighPointSum" θα πρέπει πρώτα να πάρουμε το πρώτο Rdd "nodePairRddFirst" και να το φέρουμε στη μορφή <[node0,node1],0L>, ώστε να μπορέσουμε να το κάνουμε join με το Rdd " pCombinationPair" το οποίο περιέχει όλα τα δυνατά ζευγάρια και τελικά να πάρουμε μόνο τα ζευγάρια των γειτόνων ενός κόμβου τα οποία υπάρχουν στην πραγματικότητα.

```
JavaPairRDD<Tuple2<Long, Long>, Long> nodePairRddT = nodePairRddFirst.mapToPair(s -> {

   return new Tuple2<>(new Tuple2<>(s._1, s._2), 0L);
});

JavaPairRDD<Long, Double> nodeNeighPointSum =
      pCombinationPair.join(nodePairRddT).mapToPair(s -> {

         return new Tuple2<>(s._2._1, 1.0);
      }).reduceByKey((a, b) -> {
         return a + b;
      });
```

```
(3558,287.0)
(1084,73.0)
(3586,453.0)
(1410,145.0)
(3456,602.0)
(3702,374.0)
(3272,394.0)
(3066,55.0)
(4038,20.0)
(1894,1051.0)
```

Εικόνα 7

Αφού έχουμε δημιουργήσει τα 2 Rdds τα οποία μας δίνουν τον αριθμητή και παρονομαστή τους κλάσματος clustering coefficient για κάθε κόμβο, μπορούμε να το υπολογίσουμε. Το Rdd που θα δημιουργηθεί θα μας δείχνει το clustering coefficient για κάθε κόμβο και θα είναι της μορφής <node,sum>:

```
JavaPairRDD<Long, Double> unClusterCoeffNode =
      nodeNeighPointSum.join(calcuNodeDegreeDen).mapToPair(s -> {

         Double sum = s._2._1 / s._2._2;

         return new Tuple2<>(s._1, sum);
      });
```

```
(384,0.8333333333333334)
(1084,0.6952380952380952)
(1410,0.6904761904761905)
(3456,0.47215686274509805)
(3702,0.5320056899004267)
(3272,0.4164904862579281)
(3066,0.6043956043956044)
(4038,0.5555555555555556)
(1894,0.5381464413722479)
(466,0.9444444444444444)
```

Εικόνα 8

2.5. Υπολογισμός global clustering coefficient

Τέλος, θα πρέπει να υπολογίσουμε το πλήθος των κόμβων που υπάρχουν στο dataset και να υπολογίσουμε το Global Clustering Coefficient το οποίο είναι το άθροισμα clustering coefficient ανά κόμβο / το πλήθος των κόμβων.

```
Long sizeOfRank = adjacencyList.count();
Double sizeOfRankD = Double.parseDouble(sizeOfRank.toString());

JavaPairRDD<String, Double> globalClusCoeff = unClusterCoeffNode.mapToPair(s -> {

   return new Tuple2<>("all_Nodes", s._2);
}).reduceByKey((a, b) -> {
   return (a + b);
}).mapToPair(s -> {

   Double globalSum = s._2 / sizeOfRankD;

   return new Tuple2<>("GlobalClusteringCoefficient --> ", globalSum);
});
```

```
(GlobalClusteringCoefficient --> ,0.6055467186200862)
```

Εικόνα 9

Global Clustering Coefficient

- Social circles: Facebook = 0,6055
- Pennsylvania road network = 0.0465

3. Παραδείγματα Εκτέλεσης Κώδικα

Αφού έχουμε ξεκινήσεις το Hadoop και το Spark πρέπει να ακολουθήσουμε με την σειρά τις παρακάτω εντολές :

- Δημιουργία φακέλων στο σύστημά μας, από τους οποίους θα διαβάσει τα input δεδομένα[1].
  - hadoop/bin/hadoop fs -mkdir -p /clustering-coefficient/facebook
  - hadoop/bin/hadoop fs -mkdir -p /clustering-coefficient/roadNet
  - hadoop/bin/hadoop fs -mkdir -p /clustering-coefficient/example

[1] Όλες οι εντολές είναι x3 διότι κάναμε δοκιμές με 3 διαφορετικά datasets.

- Προσθήκη δεδομένων από το τοπικό μας μηχάνημα στο server.
    - hadoop/bin/hadoop fs -put /vagrant/data/spark/input/facebook/* /clustering-coefficient/facebook/input
    - hadoop/bin/hadoop fs -put /vagrant/data/spark/input/roadNet/* /clustering-coefficient/roadNet/input
    - hadoop/bin/hadoop fs -put /vagrant/data/spark/input/example/* /clustering-coefficient/example/input
- Εντολή για να δούμε αν έχουν φτιαχτεί οι φάκελοι μας.
    - hadoop/bin/hadoop fs -ls /clustering-coefficient/facebook/input
    - hadoop/bin/hadoop fs -ls /clustering-coefficient/roadNet/input
    - hadoop/bin/hadoop fs -ls /clustering-coefficient/example/input
- Εντολή για να τρέξουμε το πρόγραμμά μας αφού πρώτα έχει γίνει install από το maven.
    - spark/bin/spark-submit --class org.spark.clustering.coefficient.ClusCoe /vagrant/data/spark/clustering-coefficient/target/clustering-coefficient-1.0.jar hdfs://localhost:54310/clustering-coefficient/facebook/input hdfs://localhost:54310/clustering-coefficient/facebook/output.
    - spark/bin/spark-submit --class org.spark.clustering.coefficient.ClusCoe /vagrant/data/spark/clustering-coefficient/target/clustering-coefficient-1.0.jar hdfs://localhost:54310/clustering-coefficient/roadNet/input hdfs://localhost:54310/clustering-coefficient/roadNet/output.
    - spark/bin/spark-submit --class org.spark.clustering.coefficient.ClusCoe /vagrant/data/spark/clustering-coefficient/target/clustering-coefficient-1.0.jar hdfs://localhost:54310/clustering-coefficient/example/input hdfs://localhost:54310/clustering-coefficient/example/output.
- Αντιγραφή των παραγόμενων φακέλων από το server στο τοπικό μας μηχάνημα
    - cp -r adjacencyList calcuNodeDegreeDen globalClusCoeff nodeNeighPointSum nodePairRddFirst pCombinationPair unClusterCoeffNode /vagrant/data/spark/OutputData/facebook.
    - cp -r adjacencyList calcuNodeDegreeDen globalClusCoeff nodeNeighPointSum nodePairRddFirst pCombinationPair unClusterCoeffNode /vagrant/data/spark/OutputData/roadNet
    - cp -r adjacencyList calcuNodeDegreeDen globalClusCoeff nodeNeighPointSum nodePairRddFirst pCombinationPair unClusterCoeffNode /vagrant/data/spark/OutputData/example
- Αντιγραφή output φακέλου μέσα από το dfs
    - hadoop/bin/hadoop fs -cp -r /clustering-coefficient/roadNet/output
    - hadoop/bin/hadoop fs -cp -r /clustering-coefficient/facebook/output
    - hadoop/bin/hadoop fs -cp -r /clustering-coefficient/example/output