HW4-PageRank

Student ID: r09921a10

Name: 鄭翔予

- Task Description
  - Input: Given a big matrix M. Specifically the column-normalized adjacency matrix where each column represents a webpage (vertex) and where it links to the non-zero entries. M is described in p2p-Gnutella04.txt
  - Output: Calculates Google Matrix A with PageRank equation, where B = 0.8. The output is stored in output.txt and the format is (page_id, pagerank)
    - Google Matrix A:

$$A = \beta M + (1 - \beta)\left[\frac{1}{N}\right]_{N \times N}$$

    - PageRank equation:

$$r_j = \sum_{i \to j} \beta \frac{r_i}{d_i} + (1 - \beta)\frac{1}{N}$$

    - Recursive problem: r=A*r

- If M contains dead-ends, need to renormalize r^{new}

$$\forall j: r_j^{new} = r'^{new}_j + \frac{1-S}{N} \quad \text{where: } S = \sum_j r'^{new}_j$$

- Process
  1. Read input
  2. Declare two arrays in_link[] and out_link[], to help us determine whether the node has in_link or out_link.
  3. Use mapper1 to construct the rdd with format (from_id, to_id). Since there could be lots different of to_id with the same from_id, use groupByKey to combine all of them.
  4. Use another map function to set the pagerank, so now our rdd's format looks like (from_id, pagerank, [to_id1, to_id2, …]).
  5. Use mapper2 to make the data inside rdd include out_degree with format (from_id, pagerank, out_degree, [to_id1, to_id2, …]). Now we have finished our preprocess.
  6. Use mapper3 to calculate B(0.8)*r_i^{old}/d_i and summarize

by reduceByKey. Remember to set the result 0 if in-degree of j is 0 (use in_link[] array to check).

7. Calculate S by summing up all r'^{new}_j, use rdd. values().sum().
8. Use S to modify pagerank by adding (1-S)/N, where N = 10876.
9. Then, we need to update the old rdd's pagerank to the new one. However since the original rdd's format has three values each key while the new rdd only has one value each key. We first use mapper4 to format the old rdd and then leftouterjoin the two rdds.
10. The result will preserve the keys that exist in the old rdd but abort the keys that exist only in the new rdd. The key was aborted because it's out_degree = 0. Therefore we don't need to memorize the value, it won't contribute to the calculation in the next round.
11. Use mapper 5 to update the pagerank and only preserve the nodes whose out_degree != 0, at the same time formats the data so that it could fit to the start of the

next round.

12. Loop 6~11 20 times. However the last round step right after step 8.
13. Sort the answers and take the top 10 pagerank's id and write it into the file.

- Mapper
  - mapper1:

    read file and transform into the format (from1, to1), …
  - map(lambda x : (x[0], 1/N if in_link[int(x[0])] else 0, list(x[1]))):

    add initial pagerank. Set 0 if the in_degree is 0
  - mapper2:

    add out_degree
  - mapper3:

    calculate r_j'
  - mapValues(lambda x: x + (1-S)/N):

    update pagerank by addins (1-S)/N
  - mapper4:

    make the original rdd to suit the format where each key

consist only one value

- mapper5:

  make the new rdd to suit the format of the start of next round

- Reducer
  - groupByKey:

    combine all the different to_ids to the same from_id

  - reduceByKey(lambda x,y: x+y):

    sum up all the same r_j'

  - leftOuterJoin:

    preserve the original keys and abort the new keys since they don't have out_links

- Resource
  - https://github.com/stanley101music/Spark