Big Data Analytics Project Spring 2017, Option A Analyzing ACM Citation Network

1. ACM Citation Dataset

ACM citation data is extracted from DBLP, ACM, and other sources. The dataset is available on this website (https://aminer.org/billboard/citation). We use ACM-Citation-network V8 which consists of 2,381,688 papers and 10,476,564 citation relationships. After downloading the dataset, extract and look at the file "citation-acm-v8.txt" This file is the input to your program.

Each line in the "citation-acm-v8.txt" starting with a specific prefix indicates an attribute of the paper. More specifically,

#* paperTitle
#@ Authors
#t Year
#c publication venue
#index 00---- index id of this paper
#%the id of references of this paper (there are multiple lines, with each indicating a reference)
#! --- Abstract

The following is an example:

#*Information geometry of U-Boost and Bregman divergence
#@Noboru Murata,Takashi Takenouchi,Takafumi Kanamori,Shinto Eguchi
#t2004
#cNeural Computation
#index436405
#%94584
#%282290
#%605546
#%620759
#%564877
#%564235
#%594837
#%479177
#%586607
#!We aim at an extension of AdaBoost to U-Boost, in the paradigm to build a stronger classification machine from a set of weak learning machines. A geometric understanding of the Bregman divergence defined by a generic convex function U leads to the U-Boost method in the framework of information geometry extended to the space of the finite measures over a label set. We propose two versions of U-Boost learning algorithms by taking account of whether the domain is restricted to the space of probability functions. In the sequential step, we observe….

This means that the paper titled “information geometry of U-Boost and Bregman divergence” is written by Noboru Murata,Takashi Takenouchi,Takafumi Kanamori, and Shinto Eguchi
In 2004. The paper’s index is 436405 and this paper has referenced (i.e., linked to) papers with indices (94584, 282290, 6055446,….) The last paragraph starting with #! is the abstract of the paper.

Step 1. Building ACM Citation Network

The first thing you need to do is to extract the citation graph from citation-acm-v8.tx file. You want to produce a dataset that contains records of the following form:

Paper index 1 paper index 2

Where paper index 1 has cited (linked to) paper index 2. This dataset is typically called a “link graph”. How can you extract the link graph from the input file? Well, there are multiple ways to do it. The easiest way is to use customized record delimiter in hadoop TextInputFormat and then use either MapReduce or Spark to process each record and extract the citations.
When you use TextInputFormat in MapReduce or sc.TextFile in spark, the record delimiter by default is the new line character i.e., every record is a line of input. For this project, we want a record to be **all the lines regarding one paper. Since each paper starts with “#*”, we can customize the record delimiter to be“#*” instead of the new line character.**
That way everything between “#*” and the next “#*” in the input file is stored in a single record.

1.1 Building the citation Network using MapReduce

To customize the record delimiter in MapReduce, you can add the following lines to your driver class:

```
Configuration conf = getConf();
conf.set("textinputformat.record.delimiter", "#*");
Job job = new Job(getConf());
```

With this configuration, your map function will receive a chunk of data for a paper starting with the paper title. For example,

Information geometry of U-Boost and Bregman divergence
#@Noboru Murata,Takashi Takenouchi,Takafumi Kanamori,Shinto Eguchi
#t2004
#cNeural Computation
#index436405
#%94584
#%282290
#%605546

#%620759
#%564877
#%564235
#%594837
#%479177
#%586607
#!We aim at an extens

Then you need to parse this chunck of data in your map function, extract the paper_index and the index of all of its references and emit pairs in the following format:

Paper_index1 paper_index1

Where paper_index1 links to (cites) paper_index2

1.2 Building the citation Network using Spark

If you choose to build the citation graph in spark, you need to first open the file conf/spark-defaults.conf under your spark installation directory and add the following lines to it. **Do this in all of your machines (this allows hadoop writables to be serialized in spark):**

```
spark.serializer=org.apache.spark.serializer.KryoSerializer
spark.kryo.classesToRegister=org.apache.hadoop.io.LongWritable,org.apache.hadoop.io.Text
```

To customize the record delimiter in spark, please add the following lines to your scala program:

```
Import org.apache.hadoop.conf._
Import org.apache.hadoop.io._
Import org.apache.hadoop.mapreduce.lib.input._
@transient val hadoopConf= new Configuration
hadoopConf.set("textinputformat.record.delimiter","#*")
```

Now you can create an rdd from the input file as follows:

```
Val inputrdd= sc.newAPIHadoopFile(<Input_file>,classOf[TextInputFormat],
classOf[LongWritable], classOf[Text], hadoopConf).map{case(key,value)=>value.toString}.
filter(value=>value.length!=0)
```

After this operation every rdd element is a chunck of data for a paper similar to what you receive in your map function above if you use mapreduce. Take a few samples of rdd elements to see how they look like and then write a function to extract all the references for each paper and create an rdd in the following form:
Paper_index1 paper_index2

Where paper_index1 links to (cites) paper_index2

Note: Some papers may not have any outgoing links (that is they may not have any references starting with #%) . You can just ignore those papers and not emit anything for them.

Step 2. Performing Graph Analytics on ACM Citation Network

In this step, you are to analyze the citation graph as follows:

1- Visualize the in-degree distribution of the ACM citation network.

2- Implement a weighted page rank to find the most influential papers in the ACM citation dataset

3- Find the average clustering coefficient and average path length of the graph (Optional)

These requirements are explained in more details in the following sections:

Step 2.1 visualizing the in-degree distribution of the ACM citation network

The output of this part should be a graph which shows the in-degree-distribution of the citation network. The in-degree of a paper *p* is the number of other paper which linked to (cited) p.

The in-degree distribution p(k) of a network is the fraction of nodes in the network with degree k. That is,

$$P(k)=\frac{n_k}{n}$$

Where n_k is the number of nodes (papers) with k inlinks and n is the total number of nodes in the graph.

This part of the project is similar to the second problem in assignment 5 and you can use Spark Graphx library to get the total number of nodes and the number of in-links to each node, and calculate the in-degree distribution.

Similar to assignment 5, The x-axis in the graph should represent k (in-degree) and the y-axis should represent p(k) (fraction of nodes with in-degree k). Please use logarithmic scale for both x and y axis.

Step 2.2 Implementing the Weighted Page Rank Algorithm

The page rank algorithm measures the importance of a page by the importance of the other pages that linked to it. The page rank algorithm in general can be applied to any type of connected graph. For example, in social graphs, where the nodes are people and the edges are the friendship relation, the page rank algorithm can be used to identify most popular people. Or in the World Wide Web graph, where the nodes are websites and the edges are the links between the website, the page rank algorithm can be used to identify and rank most relevant pages.

This step is the major step of you project. In this step, you are to implement a weighted pageRank algorithm on the ACM citation network data set to identify the most influential papers. The goal is to find the top 10 papers with the highest weighted page rank in ACM citation network. The pagerank algorithm used here is based on a highly cited paper by Xing and Ghorbani [1]. As a graduate student, you should be able to read this paper, understand it and implement its proposed algorithm.

Nevertheless, I write a short summary of the weighted page in the following section.

You can implement the weighted page rank algorithm using spark core or spark sql.

Once you calculated the page rank for all papers in your link dataset, find the top 10 papers with the highest page rank. Your output must be in the following form:

paper title **the number of citations (in-links)** **and the page rank**

First try your program on a small link graph and test and debug your code. I typically like to develop my program incrementally in spark-shell and take a few samples of each rdd after performing operations on it.

Once you are confident that your program produces correct page ranks for smaller data, you can run it on your UIS cluster or EMR for the full dataset. Make sure that you clean up disk on your UIS cluster, empty your trash and remove all the old files from local disk and hdfs. Also remove the log files on hadoop/logs folder and spark/logs folder.

What is Weighted PageRank and how it is computed

Weighted page rank is an extension to the standard page rank algorithm which takes into account the importance of in-links and out-links of a page and distributes rank scores based on the popularity of a page.

More formally, Weighted PageRank is a function that assigns a real number to each node in a graph. The intent is that the higher the PageRank of a node, the more important it is. The weighted page rank *PR(u)* of a node *u* is calculated as follows [1]:

$$PR(u) = \frac{1-d}{N} + d \sum_{v \in in(u)} PR(v) \times W^{in}_{(v,u)} \times W^{out}_{(v,u)}$$

Where

- d: is a constant (it is typically set to 0.85)
- **N**: is the total number of nodes (In this case total number of papers in the citation network)
- **in(u):** The set of all the nodes which link to node **u** (In this case, the set of all papers which cited u)
- $w^{in}_{(u,v)}$ **:** is an *in-weight* of link(v, u). It is calculated as the ratio of the number of in-links (incoming links) to node *u* over the number of in-links to all references of node *v.*

$$W^{in}_{(v,u)} = \frac{\#inlinks(u)}{\sum_{k \in out(v)} \#inlinks(k)}$$

Where, out(v) means all the reference nodes of v (all nodes to which v links)

- $w^{out}_{(u,v)}$: *out-weight* of link(v, u) . It is calculated as the ratio of the number of out-links

  (outgoing links) from node *u* and the number of out-links from all references of node *v:*

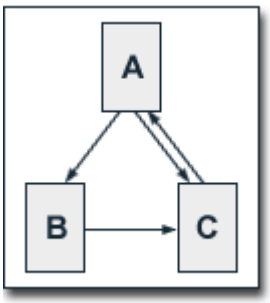
$$w^{out}_{(u,v)} = \frac{\#outlinks(u)}{\sum_{k\in} out(v)\#outlinks(k)}$$

Where, out(v) means all the reference nodes of v (all nodes to which v links)

At the initial point, each node is initialized with PageRank 1/N and the sum of the PageRank is 1. **For this project, you only need to do 10 iterations** and compute the page ranks for each paper after the 10th iteration.

Example

We regard a small citation network consisting of three papers A, B and C, whereby paper A cited paper B and C, paper B cited paper C and paper C cited paper A. The following figure illustrates the link graph of this simple problem.



We calculate the page rank for each node in the following steps:

1) **Initialization**
   d=0.85, N=3, PR(A)=1/3, PR(B)=1/3, PR(C)=1/3

2) Calculating W^{in} and W^{out} for all the links:

- $W^{in}_{(A,B)} = \frac{\#inlinks(B)}{\#inlinks(B)+\#inlinks(C)} = \frac{1}{1+2} = \frac{1}{3}$, $W^{out}_{(A,B)} = \frac{\#outlinks(B)}{\#outlinks(B)+\#outlinks(C)} = \frac{1}{1+1} = \frac{1}{2}$
- $W^{in}_{(A,C)} = \frac{\#inlinks(C)}{\#inlinks(B)+\#inlinks(C)} = \frac{2}{1+2} = \frac{2}{3}$, $W^{out}_{(A,C)} = \frac{\#outlinks(C)}{\#outlinks(B)+\#outlinks(C)} = \frac{1}{1+1} = \frac{1}{2}$

- $W_{(B,C)}^{in} = \frac{\#inlinks(C)}{\#inlinks(C)} = \frac{2}{2} = 1 \quad , \quad W_{(B,C)}^{out} = \frac{\#outlinks(C)}{\#outlinks(C)} = \frac{1}{1} = 1$
- $W_{(C,A)}^{in} = \frac{\#inlinks(A)}{\#inlinks(A)} = \frac{1}{1} = 1 \quad , \quad W_{(C,A)}^{out} = \frac{\#outlinks(A)}{\#outlinks(A)} = \frac{1}{1} = 1$

3) Iteration 1:

- Calculating PR(A): C is the only node which links to A. Hence,

$$PR(A) = \frac{1-d}{N} + d * PR(C) * W_{(C,A)}^{in} * W_{(C,A)}^{out} = \frac{1-0.85}{3} + 0.85 * \frac{1}{3} * 1 * 1 = 0.3333$$

- Calculating PR(B): A is the only node that links to B. Hence,

$$PR(B) = \frac{1-d}{N} + d * PR(A) * W_{(A,B)}^{in} * W_{(A,B)}^{out} = \frac{1-0.85}{3} + 0.85 * \frac{1}{3} * \frac{1}{3} * \frac{1}{2} = 0.09722$$

- Calculating PR(C): A and B link to C. Hence,

$$PR(C) = \frac{1-d}{N} + d * \left[PR(A) * W_{(A,C)}^{in} * W_{(A,C)}^{out} + PR(B) * W_{(B,C)}^{in} * W_{(B,C)}^{out}\right]$$
$$= \frac{1-0.85}{3} + 0.85 * (\frac{1}{3} * \frac{2}{3} * \frac{1}{2} + \frac{1}{3} * 1 * 1) = 0.42777$$

4) Iteration 2:

- Calculating PR(A): C is the only node which links to A. Hence,

$$PR(A) = \frac{1-d}{N} + d * PR(C) * W_{(C,A)}^{in} * W_{(C,A)}^{out} = \frac{1-0.85}{3} + 0.85 * 0.42777 * 1 * 1 = 0.4136045$$

- Calculating PR(B): A is the only node that links to B. Hence,

$$PR(B) = \frac{1-d}{N} + d * PR(A) * W_{(A,B)}^{in} * W_{(A,B)}^{out} = \frac{1-0.85}{3} + 0.85 * 0.3333 * \frac{1}{3} * \frac{1}{2} = 0.0972175$$

- Calculating PR(C): A and B link to C. Hence,

$$PR(C) = \frac{1-d}{N} + d * \left[PR(A) * W_{(A,C)}^{in} * W_{(A,C)}^{out} + PR(B) * W_{(B,C)}^{in} * W_{(B,C)}^{out}\right]$$
$$= \frac{1-0.85}{3} + 0.85 * (0.3333 * \frac{2}{3} * \frac{1}{2} + 0.09722 * 1 * 1) = 0.227072$$

5) Repeat the same computation for 8 more iterations.

Step 2.3 (Optional +10 bonus) Finding Average Clustering Coefficient and Average Path Length of the network and compare it to a random graph:

2.3.1 What is a Small-World Network?

Many real-world networks including social networks, Wikipedia link network, Internet architecture network, food chains, electric power grids, network of brain neurons, network of connected proteins, etc. exhibit properties of a type of graph called "Small-World" networks [2]. Small-world networks have two main properties:

1. Most of the nodes in the network have a small degree which means most of the nodes are not neighbors of one another, but for every given node, a high fraction of its neighbor's o are also neighbors of each other. This property is typically measured using **Average Clustering Coefficient** of the network.
2. Despite the small degree and dense clustering of most nodes in the graph, it is possible to reach any node in the network from any other network relatively quickly by traversing a small number of edges. This property is typically measured using **Average Shortest Path Length of the network**.

If you are more interested to know about small-world graphs, you can watch this short Youtube video: https://www.youtube.com/watch?v=QUWds9gt6aE

2.3.2 Computing Average Clustering Coefficient

This step is optional. If you have time and would like to build more on your spark skills, then try to do this part as well.

A graph is called *complete* if every vertex is connected to every other vertex. In a given graph we might have many subsets of nodes that are complete. These complete sub-graphs are called *cliques*. A large number of cliques in a graph indicates that the graph has a locally dense structure which could indicate that it is possible to reach any node from any other node in the network relatively quickly by traversing a small number of edges (https://en.wikipedia.org/wiki/Small-world_network)

Unfortunately, finding the number of cliques in a graph is computationally very hard. Even to determine if a graph has a clique or not turns out to be an NP- Complete Problem. Local clustering coefficient, introduced by Watts and Strogatz in their seminal paper [2] can give us a feel of local density of the graph without computational cost of finding all the cliques. The local clustering coefficient of a node v in a graph is defined as follows:

$$cc(v) = \frac{t_v}{2k_v(1-k_v)}$$

Where t_v is the number of triangles that contains v and k_v is the degree of node *v*. A triangle is simply three nodes that are connected to each other (a complete graph of three nodes). For an example and tutorial on clustering coefficient, please watch this short tutorial video

(https://www.youtube.com/watch?v=K2WF4pT5pFY) In this video notion N_v is used instead of t_v to refer to the number of triangles which contain v but they are essentially the same thing.

You can use Graphx package to find the triangle count for each vertex (i.e., t_v). Graphx also has a "degrees" method which gives the degree of each vertex.

Having the triangle count at each node and its degree, you can easily compute the local clustering coefficient at each vertex.

Once you compute the local clustering coefficient at each vertex, take the average of clustering coefficients for all the nodes and write this average value into a file.

2.3.3 Computing Average Path Length

The average path length of a graph is the average length of the shortest path between every two nodes in the graph and is computed as follows:

$$\text{Average\_path\_length} = \frac{\sum_{i \neq j} d(vi,vj)}{n(n-1)}$$

Where *d(vi,vj)* is the shortest path from *vj* to *vj*

You can find the shortest path between vertices in spark using ShortestPath object from spark graphx library : (http://spark.apache.org/docs/latest/api/scala/index.html#org.apache.spark.graphx.lib.ShortestPaths$)

2.3.3 Comparing the average clustering coefficient and average path length to a random graph

To determine if the ACM citation network is a small world network, you need to create a random graph with the same number of nodes and compare the average path length and clustering coefficient of the ACM citation graph with that of the random graph. If the clustering coefficient of the ACM citation graph is significantly greater than the clustering coefficient of the random graph, then the ACM citation graph exhibits small world network properties.

To create a random graph with the same number of nodes, you can take every two nodes in the graph and create a link between them with a probability *p* (for example, you can set *p*=0.5) . Typically, instead of comparing against a single random graph, the clustering coefficient and average path length for the target graph is compared against an **ensemble** of random graphs. In other words, you create a collection of random graphs, compute their clustering coefficients and path lengths, take their averages, and compare these averages to the clustering coefficient and path length of the ACM citation graph.

2. What you need to turn in:

Here are the items that you need to submit. If the submission is too big, please upload it to your box account and share the folder with me.

1. **"Link.txt ".** The citation graph you built in step 1. The records in this dataset should be in the following format.
   Paper_index 1 paper_index2

   Where paper_index 1 links to paper_index2. If your spark or MapReduce program, produced multiple files for the citation graph, just compress them and submit the zip file.

2. **The in-degree distribution graph you produced in step 2.2**

3. **"paper_rank.txt". The top 10 papers with the highest page rank you produced from step 2.2.** The records of this data set should be in the following format (fields are tab separated):
   Paper_title number_of_in-links page_rank

4. If you did step 2.3,you need to submit your script for computing average clustering coefficient and average path length of the ACM citation graph as well as an ensemble of random graphs. Then in your report you should compare the clustering coefficient and path length of the ACM graph with that of the ensemble of random graphs and discuss whether the ACM citation shows small-world network properties.

5. **"Source.zip".** All your source files in a zip folder along with a readme.txt describing the submission
6. **Project Report (Doc or PDF File) with the following format:**
   - Project title & your name
   - Introduction: A short paragraph explaining the
   - Methodology: explaining the algorithms that you used (your MapReduce and spark programs with example)
   - Results and Conclusion
   - References: if you used any website, book, etc. please list it here and cite it in your text.

   Your report must be well-written, clear and grammatically correct.

7. **Presentation:** You need to prepare a set of slides and present your work. This should include 20-25 slides. The first slide should include the title of your project and your name. The next slides should include a quick overview of the presentation. The rest of the slides should explain your work by example and interpret the results.

3. Grading Rubric:

Step 1- Building Citation Network	**20%**
Step2- Computing Weighted Page Rank	**65%**
Report and Presentation	**15%**

References:

[1] W. Xing and A. Ghorbani, "Weighted PageRank algorithm," *Communication Networks and Services Research, 2004. Proceedings. Second Annual Conference on*, 2004, pp.305-314.

[2] D. J. Watts and Steven Strogatz, "Collective dynamics of 'small-world' networks". *Nature*. **393**(6684), 1998, PP.440–442.