|  |  |
| --- | --- |
| Big Data Analytics Project FALL 2016 - Analyzing ACM Citation Network | Submitted By: Marakhi Der, Karan Kanwar |

**TABLE OF CONTENTS**

* Introduction…………………………………………………… 3

In-degree distribution and weighted page rank for ACM citation dataset

* Detail description of the project……………………… 4

Theoretical explanation about weighted-page rank algorithms, in-degree distribution algorithm and overview about data set.

* technical layout of the project……………………… 7

Explaining the methodology used for generate required output with examples of code implementation.

* Results and Conclusion…………………………………... 13

.

**Introduction**

This project is using ACM citation data set which is extracted from DBLP, ACM, and other sources. This project is consisting of three steps. Step 1 for building the citation network. Step 2 for visualizing the in-degree distribution of the ACM citation network. Step 3 for Implementing the Weighted Page Rank Algorithm.

* Step 1: Building the citation network

The input dataset of this project is containing semi-structured data. In order to fetch all required information, various filtering and mapping technique used.

* Step 2: visualizing the in-degree distribution

Step 2 is the next important stage of the project to deal with is structured data which is obtain from step 1. In this step we will generate in-degree distribution graph using Spark’s GraphX library. Graph will be consisting of The x-axis which represent k (in-degree) and the y-axis which represent p(k) (fraction of nodes with in-degree k)

* Step 3: Weighted Page Rank Algorithm

Final step of the project is about implementing Weighted Page Rank Algoritham, developed by W. Xing and A. Ghorbani. This step will measure the importance of a page by the importance of the other pages that linked to it. At the end, this step will generate top 10 papers by its page rank.

**Detail description of the project**

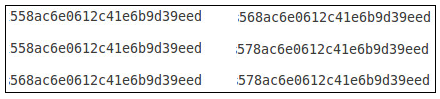
The ACM citation data-set is consisting of various attributes (like papertitle, authors, year of publish, publication venue, paper index, reference paper index) separated by specific prefix (‘#\*’ for paper title, ‘#@’ for author, ‘#t’ for year etc). Below is the example file layout.

* Input File Format

|  |
| --- |
|  |

Out of these many attributes, only few attributes are of importance, for this project, so remove other attributes. Important Attributes are Title, Paper Index and Reference paper index. For building ACM citation network Paper index and Reference Paper Index required. The Title will be used in last step of this project which is output of weighted page rank algorithm. Output file of ACM citation network based on above input file is shown below.

* Output File



* GraphX

Output of Step 2 of this project produce graph, Based on indegree-distribution. For this step spark’s in build GraphX library is useful. GraphX unifies ETL, exploratory analysis, and iterative graph computation within a single system. You can [view](http://spark.apache.org/docs/latest/graphx-programming-guide.html#the-property-graph) the same data as both graphs and collections, [transform](http://spark.apache.org/docs/latest/graphx-programming-guide.html#property-operators) and [join](http://spark.apache.org/docs/latest/graphx-programming-guide.html#join-operators) graphs with RDDs efficiently, and write custom iterative graph algorithms. GraphX competes on performance with the fastest graph systems while retaining Spark's flexibility, fault tolerance, and ease of use. In addition to a [highly flexible API](http://spark.apache.org/docs/latest/graphx-programming-guide.html#graph-operators), GraphX comes with a variety of graph algorithms, many of which were contributed by users.

* In-degree Distribution

The [degree](https://en.wikipedia.org/wiki/Degree_(graph_theory)) of a node in a network (sometimes referred to incorrectly as the [connectivity](https://en.wikipedia.org/wiki/Connectivity_(graph_theory))) is the number of connections or [edges](https://en.wikipedia.org/wiki/Edge_(graph_theory)#Graph) the node has to other nodes. If a network is [directed](https://en.wikipedia.org/wiki/Directed_graph), meaning that edges point in one direction from one node to another node, then nodes have two different degrees, the in-degree, which is the number of incoming edges, and the out-degree, which is the number of outgoing edges.

The degree distribution P(k) of a network is then defined to be the fraction of nodes in the network with degree k. Thus if there are n nodes in total in a network and nk of them have degree k, we have P(k) = nk/n.

Sample output of this step is shown below.



* Weighted Page Rank

This step is the last and major step of this project. In this step, through the use of weighted page Rank algorithm on the ACM citation network data set, 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 output file which is generated at the end of this step contains paper title, inlink, page rank. Below is the screen shot of sample output file.

|  |
| --- |
|  |

With the rapid growth of the Web, users get easily lost in the rich hyper structure. Providing relevant information to the users to cater to their needs is the primary goal of website owners. Therefore, finding the content of the Web and retrieving the users’ interests and needs from their behavior have become increasingly important. Web mining is used to categorize users and pages by analyzing the users’ behavior, the content of the pages, and the order of the URLs that tend to be accessed in order. Web structure mining plays an important role in this approach. Two-page ranking algorithms, HITS and PageRank, are commonly used in web structure mining. Both algorithms treat all links equally when distributing rank scores. Several algorithms have been developed to improve the performance of these methods. The Weighted PageRank algorithm (WPR), an extension to the standard PageRank algorithm. WPR takes into account the importance of both the inlinks and the outlinks of the pages and distributes rank scores based on the popularity of the pages. The results show that WPR performs better than the conventional PageRank algorithm in terms of returning larger number of relevant pages to a given query.

The more popular webpages are, the more linkages that other webpages tend to have to them or are linked to by them. The proposed extended PageRank algorithm–a Weighted PageRank Algorithm–assigns larger rank values to more important (popular) pages instead of dividing the rank value of a page evenly among its outlink pages. Each outlink page gets a value proportional to its popularity (its number of inlinks and outlinks). The popularity from the number of inlinks and outlinks is recorded as





Above mention equations will be used later, to calculate the final page rank.



**technical layout of the project**

Import all the required libraries in to the project and do configuration properly. Below is the screen shot for the same.

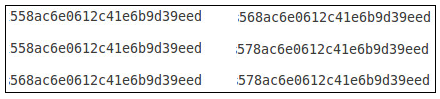
|  |
| --- |
|  |

* Filtering And Mapping

Input RDD is ready for this project to work upon.

|  |
| --- |
|  |

Split the input RDD data by “\n#!” and take its first part only, this way the attribute abstract will be removed from the data. Then Split by ‘#index’ and take 2nd part, by doing this, the attributes like Title, author, publication venue, year will be removed. It doesn’t necessary that each paper refer some other papers. So we have to handle this situation. Split RDD by “\n#%” in 2 parts only. After split check if there is/are any reference paper, and if no reference paper emits only spaces otherwise emit both part. Then filter the RDD data which contains spaces. By this time only data remain in the RDD is Paper index and reference paper index. Now to link paper index to each of its reference paper index, use flatMapValues. By doing these the remaining data is our output of step 1 which looks like as shown below.



* Graph and in-degree

|  |
| --- |
|  |

GraphX produce graph by accepting numeric data as node. But the o/p file of 1st step contains the nodes which are alphanumeric. So, there is need to convert them to numeric data. It is done by Murmurhash which convert each unique string to unique integer. Also its required to store these unique mapping link, by doing this, conversion of integer node to original string node is possible.

|  |
| --- |
|  |

Produce graph from the RDD which contain numeric node. Take in-degree for nodes. Count the total node for same in-degree. Get total node in the graph. And generate in-degree distribution.

* Weighted Page Rank

|  |
| --- |
|  |

Create 4 temporary table in-degree, out-degree, initial page rank and link. This is done by in-build hive in spark.

|  |
| --- |
|  |

Ten iterations are required as mention in the project requirement to get the final page rank. Implement function which make conversion between float and double without any issue.

To understand the query for implementation of weighted page rank, let’s take small example which consist of below tables.

Link Table

|  |  |
| --- | --- |
| In-Node | Out-Node |
| A | B |
| A | C |
| B | C |
| C | A |

In-degree Table

|  |  |
| --- | --- |
| Node | In-degree |
| A | 1 |
| B | 1 |
| C | 2 |

Out-degree Table

|  |  |
| --- | --- |
| Node | out-degree |
| A | 2 |
| B | 1 |
| C | 1 |

Inner most query which joins above mention tables and produce below output table.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| In-degree of RN | Out-degree of RN | Reference-node (RN) | In-node | Out-node(ON) | In-degree of ON | Out-degree of ON |
| 1 | 1 | B | A | B | 1 | 1 |
| 2 | 1 | C | A | B | 1 | 1 |
| 1 | 1 | B | A | C | 2 | 1 |
| 2 | 1 | C | A | C | 2 | 1 |
| 2 | 1 | C | B | C | 2 | 1 |
| 1 | 2 | A | C | A | 1 | 2 |

Group by in-node, out-node, in degree of out-node and out-degree of out node and do the summation. Remove reference node.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sum(In-degree of RN ) | Sum(Out-degree of RN) | In-node | Out-node(ON) | In-degree of ON | Out-degree of ON |
| 3 | 2 | A | B | 1 | 1 |
| 3 | 2 | A | C | 2 | 1 |
| 2 | 1 | B | C | 2 | 1 |
| 1 | 2 | C | A | 1 | 2 |

Divide In-degree of ON by Sum (In-degree of RN) and Out-degree of ON by Sum(Out-degree of RN)

|  |  |  |  |
| --- | --- | --- | --- |
| In-node | Out-node(ON) | In-degree of ON/ Sum(In-degree of RN ) | Out-degree of ON/ Sum(Out-degree of RN) |
| A | B | 1/3 | 1/2 |
| A | C | 2/3 | 1/2 |
| B | C | 2/2 | 1/1 |
| C | A | 1/1 | 2/2 |

Multiply column 3 and 4 from above table to get the final table of inner most query.

|  |  |  |
| --- | --- | --- |
| In-node | Out-node(ON) | Column3 \* column4 |
| A | B | (1/3) \*(1/2) |
| A | C | (2/3) \*(1/2) |
| B | C | (2/2) \*(1/1) |
| C | A | (1/1) \*(2/2) |

Now join the rank table with above table based on in-node.

|  |  |  |  |
| --- | --- | --- | --- |
| Rank | In-node | Out-node(ON) | Column3 \* column4 |
| 1/3 | A | B | (1/3) \*(1/2) |
| 1/3 | A | C | (2/3) \*(1/2) |
| 1/3 | B | C | (2/2) \*(1/1) |
| 1/3 | C | A | (1/1) \*(2/2) |

Now do the multiplication of column1 and column 4 in above table.

|  |  |  |
| --- | --- | --- |
| In-node | Out-node(ON) | Rank \* Column3 \* column4 |
| A | B | (1/3) \*(1/2) \*(1/3) |
| A | C | (2/3) \*(1/2) \*(1/3) |
| B | C | (2/2) \*(1/1) \*(1/3) |
| C | A | (1/1) \*(2/2) \*(1/3) |

Now group by out-node(ON) and do the summation. Remove in-node. Below is the final output from the hive query.

|  |  |
| --- | --- |
| Out-node(ON) | Rank \* Column3 \* column4 |
| B | (1/3) \*(1/2) \*(1/3) |
| C | ((2/3) \*(1/2) \*(1/3)) + ((2/2) \*(1/1) \*(1/3)) |
| A | (1/1) \*(2/2) \*(1/3) |

Convert above table into RDD and do below mention transformation. Its final wighted page rank after 1st iteration do above mention process again and again up to 10 time. After completion of each iteration update the rank table.

|  |  |
| --- | --- |
| Out-node(ON) | (Rank \* Column3 \* column4) \*D + (1-D)/N |
| B | ((1/3) \*(1/2) \*(1/3)) \*0.85 + (1-0.85)/3 |
| C | (((2/3) \*(1/2) \*(1/3)) + ((2/2) \*(1/1) \*(1/3))) \*0.85 + (1-0.85)/3 |
| A | ((1/1) \*(2/2) \*(1/3) ) \*0.85 + (1-0.85)/3 |

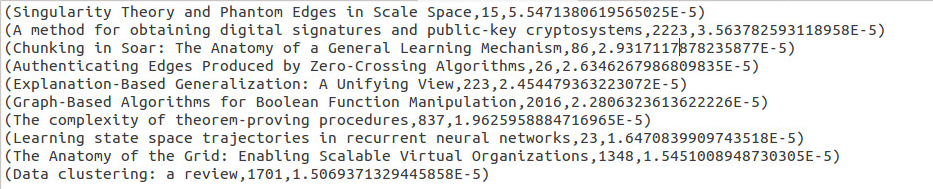
Join page rank, in link and title RDD to get the final required output. Also convert numeric node to string node through use of unique link RDD which was generated earlier. Sort the data based on the page rank. Take only top 10 rows from the data.

|  |
| --- |
|  |

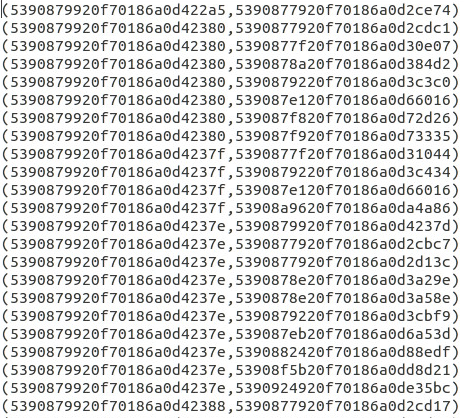
**Results and conclusion**

Below are the final result’s screen shots for each output file.

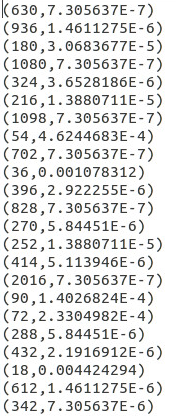
Top 10 paper with highest ranks.



Citation network graph.



In degree distribution.



**Referances**

<http://spark.apache.org/graphx/>

<https://www.wikipedia.org/>

<http://stackoverflow.com/>

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