Amna Irfan
CS 441- Hw2
10/20/2019

Employing the Map/Reduce Framework to explore the DBLP Dataset

Introduction: The map/reduce framework allows the parallel processing of datasets in which records are independent of each other. The map reduce task is deployed as a single task to the master node which then distributes the task among worker nodes called mappers and reducers. The mapper operator reads each line of the files in the input directory specified by the user and outputs key value pairs that are also determined by the user. The reducer operator reads the output from the mapper and aggregates the records based on the key. We use the map/reduce model to calculate certain statistics for the DBLP dataset that we will go in detail in the next sections.

Dataset and preprocessing: The DBLP dataset consists of information on all publicly available research papers. For example, a DBLP record will contain the following information: type of venue where the paper was public (eg,. inproceedings, book etc), the list of authors in a specific order, the year when the paper was published, etc. Since a single xml record/paper can span multiple lines, we had to do some preprocessing on the xml file to create a file that has only one record per line. We created a python script to parse xml records into a single line. The script read each line from the dblp.xml file and searched for opening and closing root tags to determine the starting and ending of a dblp record. Once an ending was detected, the record was appended to the output file. We then partitioned the dataset into 30 files. Our one-liner xml file can be found here.

Venue	# of records
Articles	2,133,033
Book	17,817
Incollection	60,069
Inproceedings	2,491,665
Master thesis	12

PHD thesis	73,973
Proceedings	42,429
www	2,385,667
Total records	7,204,665

Map/Reduce tasks: To complete this assignment's requirements, we created four different map-reduce jobs.

1. AuthorScoreMapReduce: This task calculates the publishing score of each author. The mapper reads each record and then extracts all the authors of that record/paper. For each author it calculates the authorship score such that the score for all authors for a single paper sum upto one. The mapper outputs the author's name as the key and the author's score for that paper as the value. The reducer then receives the key values pairs and aggregates/sums the scores of each author. For example, if an author has published two papers with the scores 0.3 and 0.7, his score will be 1.0. The complete output of this mapper/reduce task run on the dblp file can be downloaded from here.

2. CoauthorMapReduce: This task calculates the mean, median and max number of co-authors each author has published with (including themselves). The mapper reads each record/paper and extracts all authors. Each author is assigned the same number of coauthors for a paper. For example, if a paper has 10 authors, each author receives a score of 10. The mapper outputs the author's name as the key and the number of coauthors as the value. The reducer then reads the output of the mapper and calculates the median, mean and max number of co-authors each author published using the value from the mapper. The reducer than outputs the author's name as the key and a composite value of statistics in the following order (max|median|mean). The complete output of this mapper/reduce task run on the dblp file can be downloaded from here.

3. VenueMapReduce: This task is very similar to task 2 but instead of assigning the author as the key, it stratifies the output based on the venue as well. It uses a composite key with both the author's name and venue (for example, mark grechanik|article). The reducer does the same calculation as task 2 but with the composite key. The complete output of this mapper/reduce task run on the dblp file can be downloaded from here.

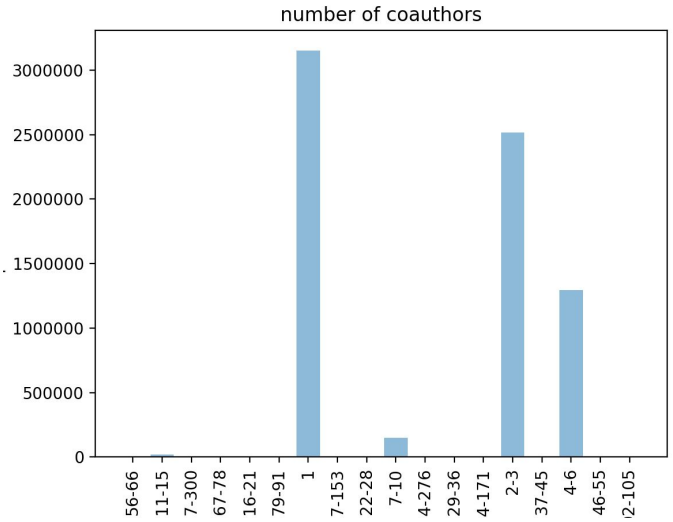
4. PublicationMapReduce: This task calculates the number of records stratified by the number of coauthors, year the paper was published in and the venue of the paper. The mapper of this task reads each paper and extracts the number of authors, the venue and

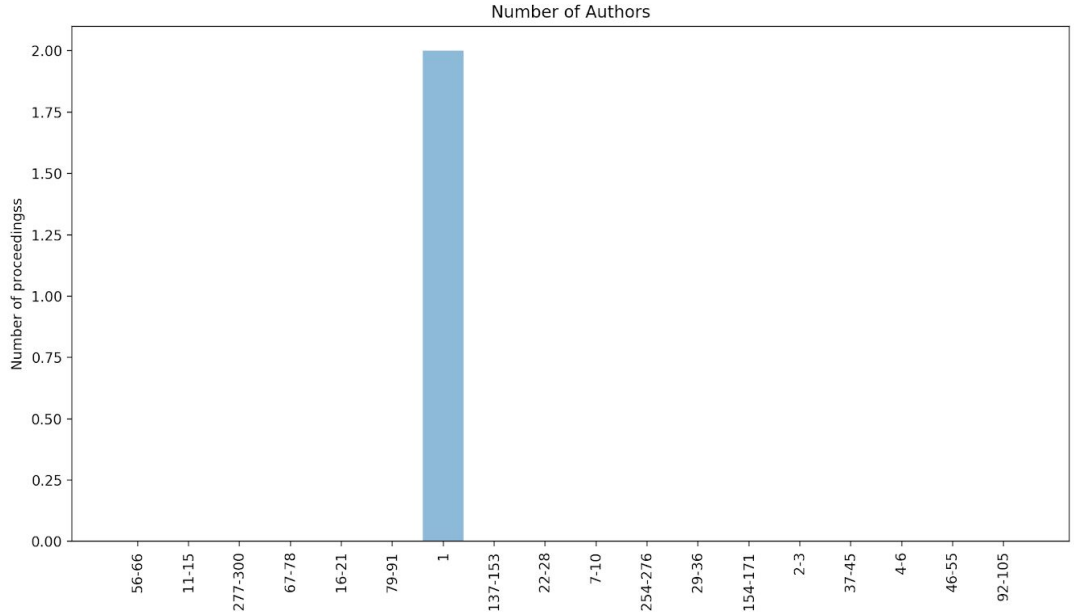
the year. It outputs a key value pair in which the key is a composite key with year, number of authors and venue concatenated (for example, 2019|article|12). The value for each key is always one as the mapper is only reading a single record at a time. The reducer than sums up all the values of the same key. We use the output of this task to create the histograms in the next section. The complete output of this mapper/reduce task run on the dblp file can be downloaded from here.

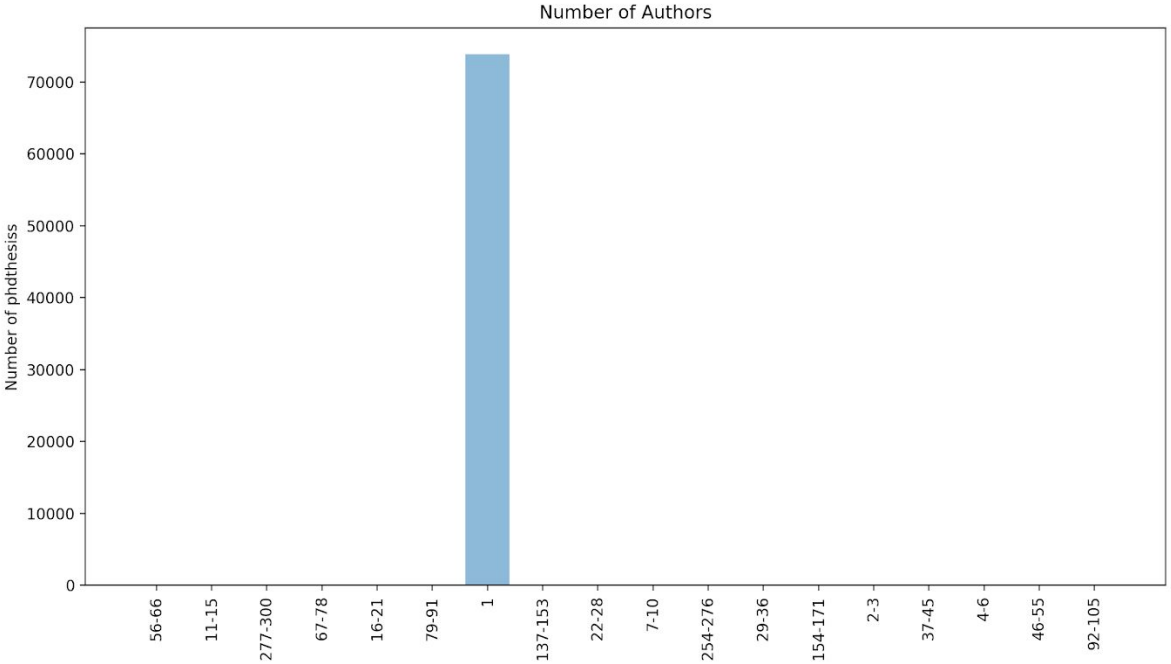
Results: We ran all the four tasks both locally, on hadoop (Hortonworks VMware) as well as AWS EMR. Instructions on running the map/reduce tasks are specified on our project's readme file. The assignments specified the following tasks:

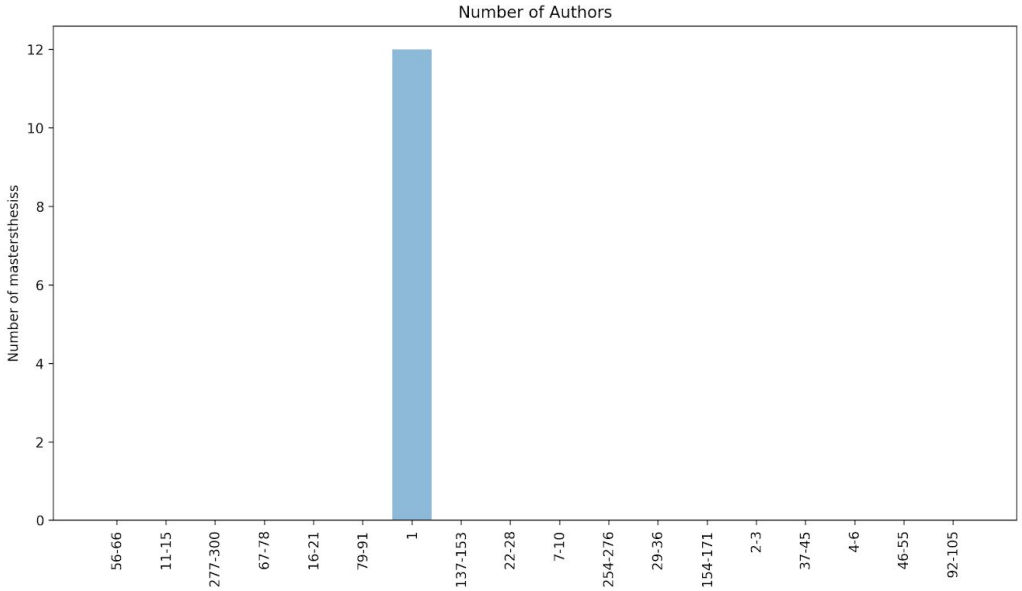
Q1. Your goal is to produce various statistics about the numbers of co-authors. One such statistics can be expressed as a histogram where each bin shows the range of the numbers of co-authors (e.g., the first bin is one, second bin is 2-3, third is 4-6, and so on until the max number of co-authors). The other statistics will produce the histogram stratified by journals, conferences, and years of publications.

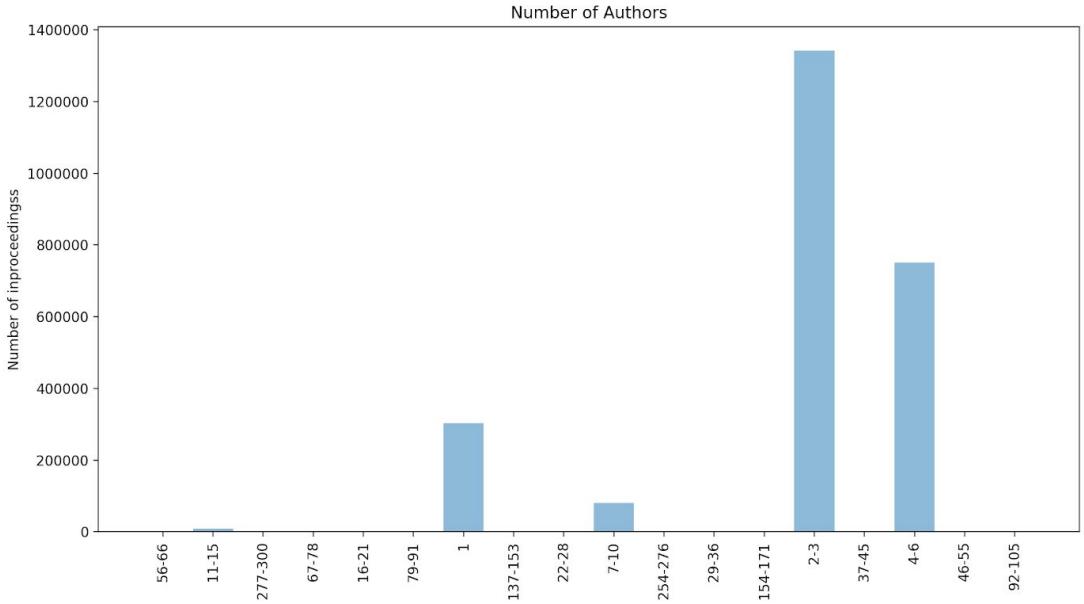
For this task, I used python's graph library called matplotlib. In order to install plotly on your computer, type in the command `python -m pip install -U matplotlib`. Make sure you have pip installed on your computer if not run the command, `sudo easy_install pip`
The python script the generates the following histograms can be found here. Run the command `python dblp_histograms.py` to get the following histograms. The first histogram is for all venues while the following are benue specific. The last histogram is based on the year when the paper was published.

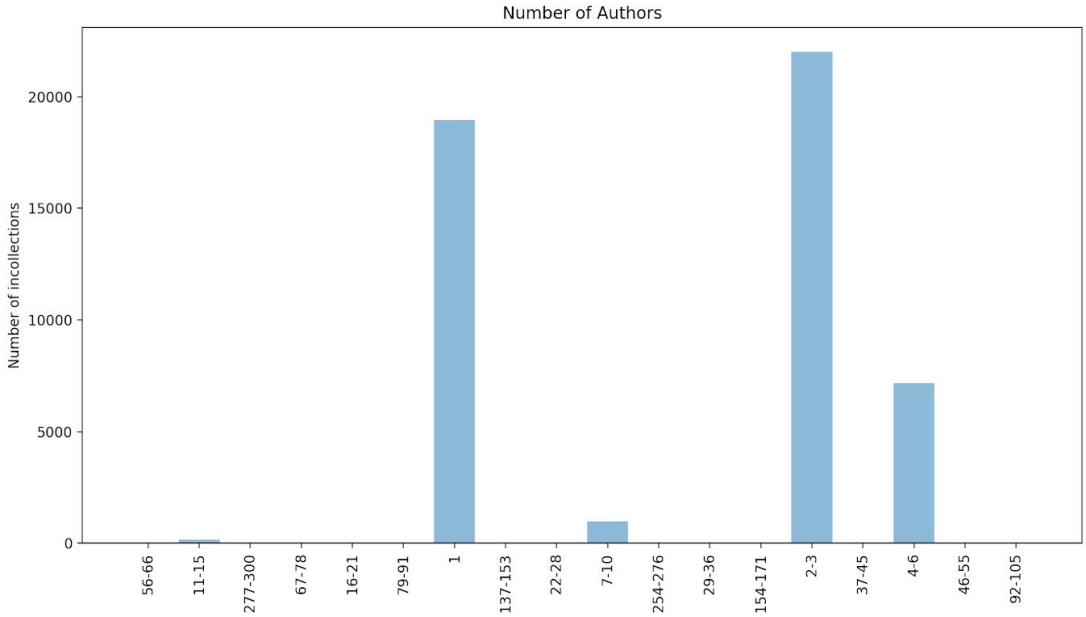


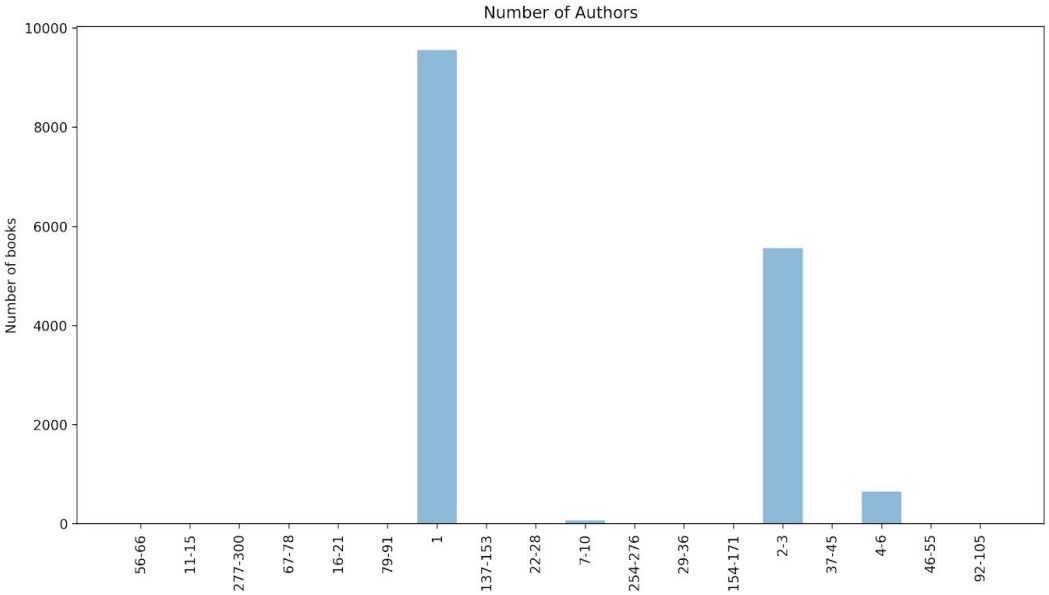

Number of Authors
Number of proceedingss
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
56-66
11-15
277-300
67-78
16-21
79-91
1
137-153
22-28
7-10
254-276
29-36
154-171
2-3
37-45
4-6
46-55
92-105

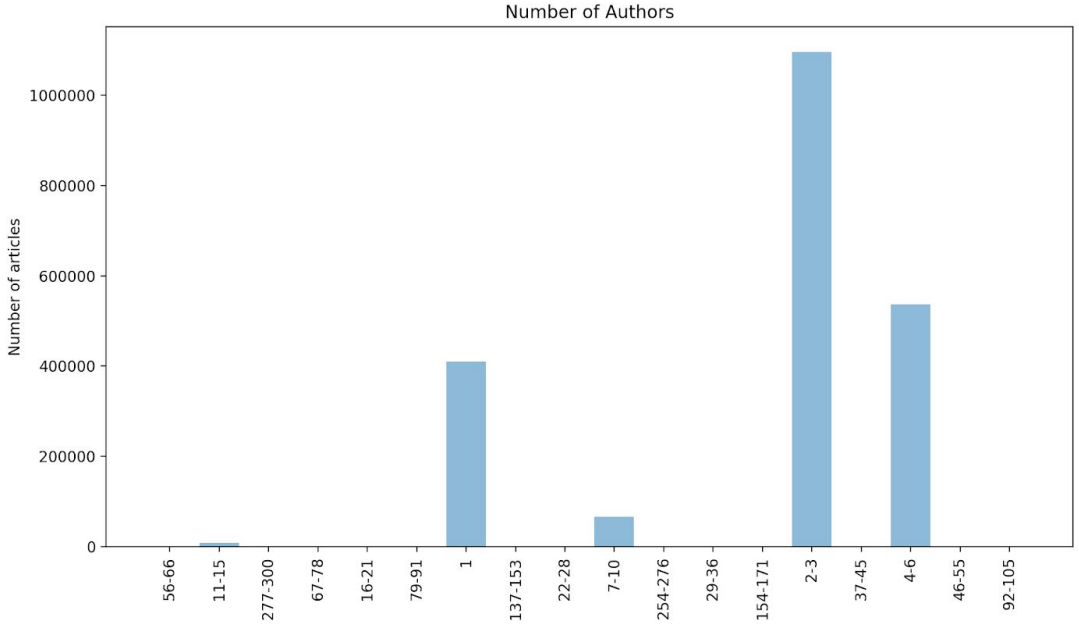

Number of Authors
Number of phdthesiss
70000
60000
50000
40000
30000
20000
10000
0
56-66
11-15
277-300
67-78
16-21
79-91
1
137-153
22-28
7-10
254-276
29-36
154-171
2-3
37-45
4-6
46-55
92-105

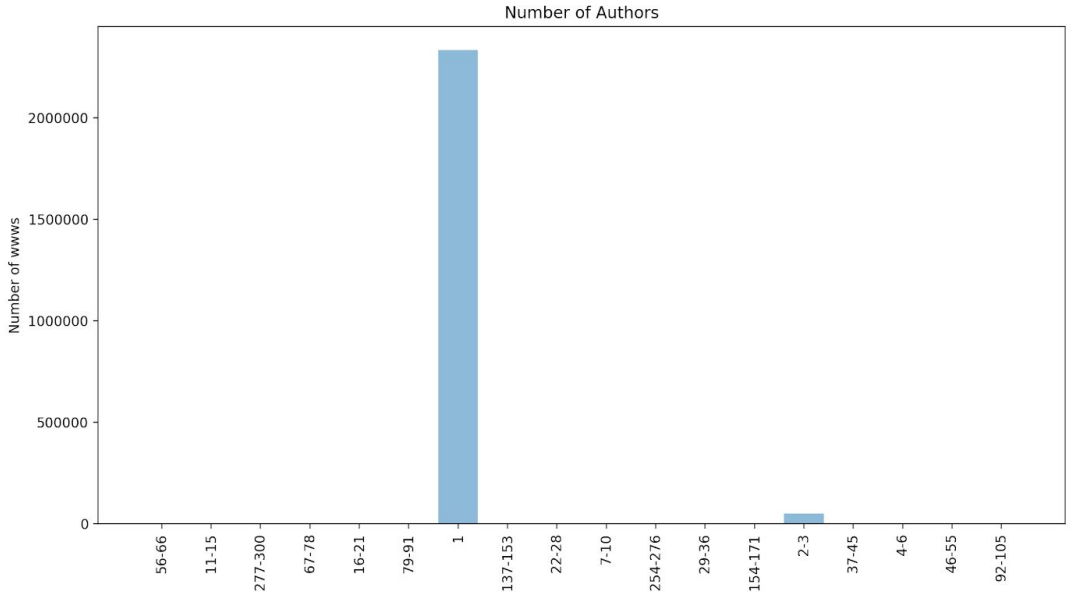

Number of Authors
Number of masterstheisis
0
2
4
6
8
10
12
56-66
11-15
277-300
67-78
16-21
79-91
1
137-153
22-28
7-10
254-276
29-36
154-171
2-3
37-45
4-6
46-55
92-105

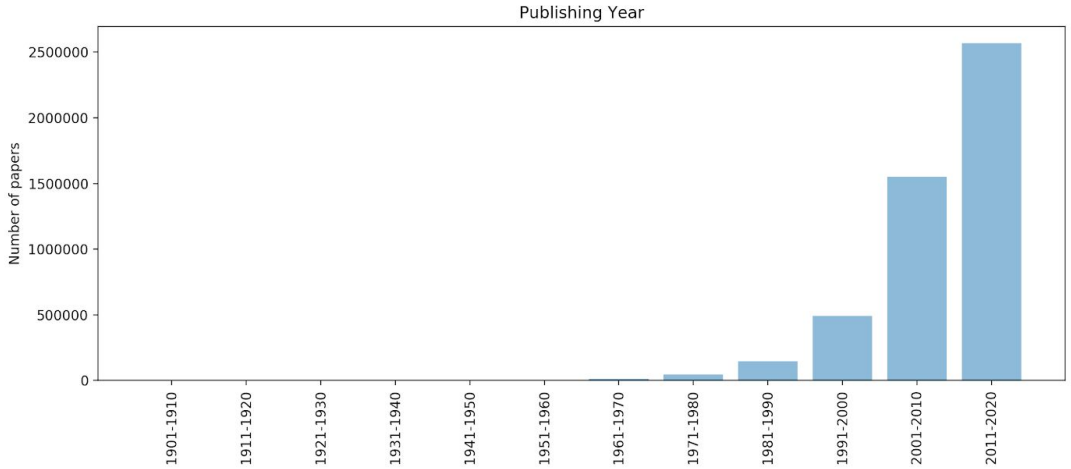

Number of Authors
Number of inproceedingss
0
200000
400000
600000
800000
1000000
1200000
1400000
56-66
11-15
277-300
67-78
16-21
79-91
1
137-153
22-28
7-10
254-276
29-36
154-171
2-3
37-45
4-6
46-55
92-105


Number of Authors
Number of incollections
20000
15000
10000
5000
0
56-66
11-15
277-300
67-78
16-21
79-91
1
137-153
22-28
7-10
254-276
29-36
154-171
2-3
37-45
4-6
46-55
92-105


Number of Authors
Number of books
10000
8000
6000
4000
2000
0
56-66
11-15
277-300
67-78
16-21
79-91
1
137-153
22-28
7-10
254-276
29-36
154-171
2-3
37-45
4-6
46-55
92-105


Number of Authors
Number of articles
1000000
800000
600000
400000
200000
0
56-66
11-15
277-300
67-78
16-21
79-91
1
137-153
22-28
7-10
254-276
29-36
154-171
2-3
37-45
4-6
46-55
92-105


Number of Authors
Number of wwws
2000000
1500000
1000000
500000
0
56-66
11-15
277-300
67-78
16-21
79-91
1
137-153
22-28
7-10
254-276
29-36
154-171
2-3
37-45
4-6
46-55
92-105



Year	Number of papers
1901-1910	0
1911-1920	0
1921-1930	0
1931-1940	66
1941-1950	156
1951-1960	2,309
1961-1970	13,403
1971-1980	46,558
1981-1990	146,748
1991-2000	492,964
2001-2010	1,549,637
2011-2020	2,567,169

Q2. Next, you will produce the list of top 100 authors in the descending order who publish with most co-authors and the list of 100 authors who publish with least co-authors. To compute the authorship score you will use the following formula. The total score for a paper is one. Hence, if an author published 15 papers as the single author without any co-authors, then her score will be 15. For a paper with multiple co-authors the score will be computed using a split weight as the following. First, each co-author receives 1/N score where N is the number of co-authors. Then, the score of the last co-author is credited 1/(4N) leaving it 3N/4 of the original score. The next co-author to the left is debited 1/(4N) and the process repeats until the first author is reached. For example, for a single author the score is one. For two authors, the original score is 0.5. Then, for the last author the score is updated 0.53/4 = 0.5-0.125 = 0.375 and the first author's score is 0.5+0.125 = 0.625. The process repeats the same way for N co-authors.

For this task, we use the authorship map/reduce output (1). The sheet with the top 100 authors who published with the most coauthors can be found inside the project. It is called *"results/top100_authors_most.csv"* . The sheet with the top 100 authors who published with the least coauthors can be found inside the project. It is called *"results/top100_authors_least.csv"* . The python script used to generate this can be found here. The top 100 authors who published with the most coauthors are the ones with the smallest authorship score and the top 100 authors who published with the least authors are the ones with the highest authorship scores.

Q3. Finally, you will output a spreadsheet where for each author you will compute the max, median, and the average number of authors for publication on which the name of the author appears. Also, you will output a stratified breakdown of these statistics by publication venues in addition to the cumulative statistics across all venues. Your job is to create the mapper and the reducer for this task, explain how they work, and then to implement them and run on the DBLP dataset. The output of your map/reduce is a spreadsheet or a CSV file with the required statistics.

For this task, we use the coauthor map/reduce output (2) and venue map/reduce output (2) . The results can be found inside the project called *"results/coauthors_stats.csv"* and *"results/coauthors_venue_stats.csv"*
The python script used to generate this can be found here.