Gregory ISLAS

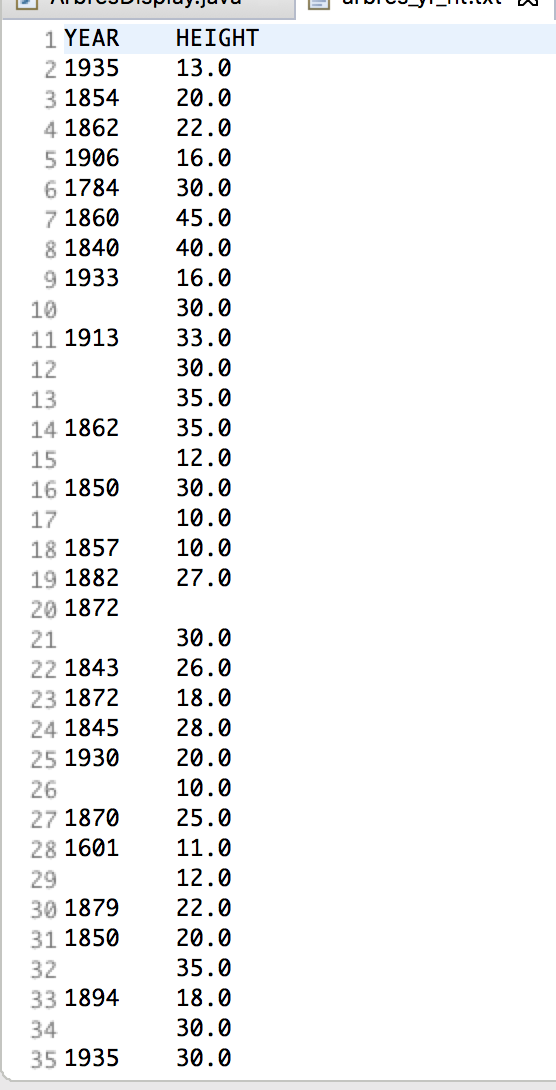
20/05/2017

Big Data Platforms Lab 1

**Brief Description/Screenshots of Programs**

**Question 2.7:**

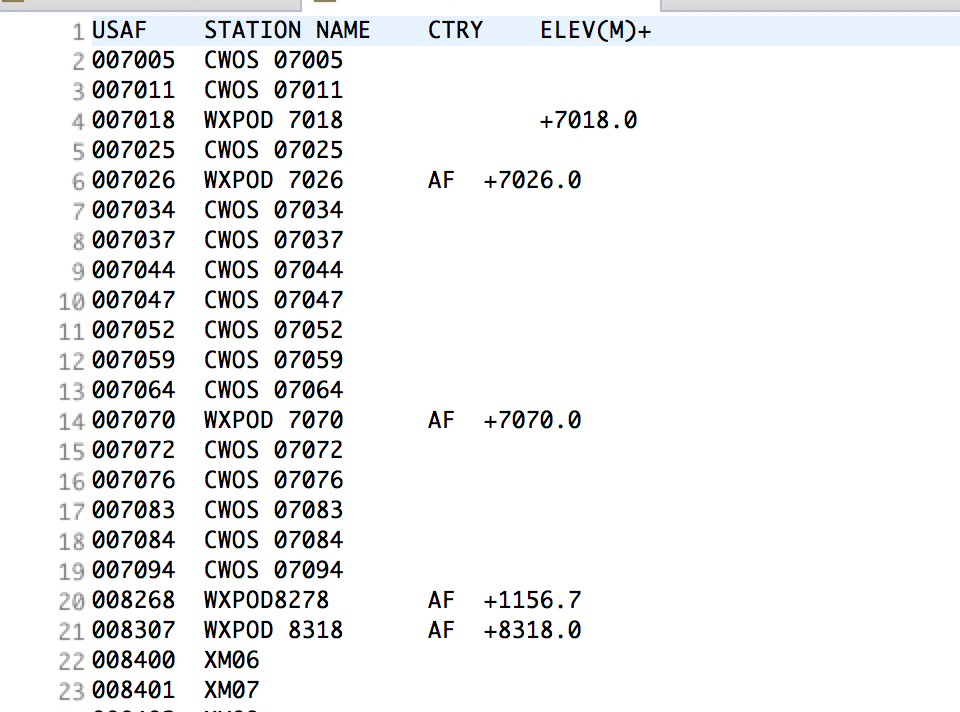
The code for question 2.7 can be found in the file **ArbresDisplay.java**. The program assumes that the file “arbres.csv” exists in a directory named “data”, which exists in the same folder that the program is in. The result of the program is placed into a file named “arbres\_yr\_ht.txt”. If the field “year” doesn’t exist in the csv file, it is replaced by a tab character in the output file. Running the program gave the following (truncated) result:



**Figure 1: Head of Output from ArbresDisplay.java**

**Question 2.8:**

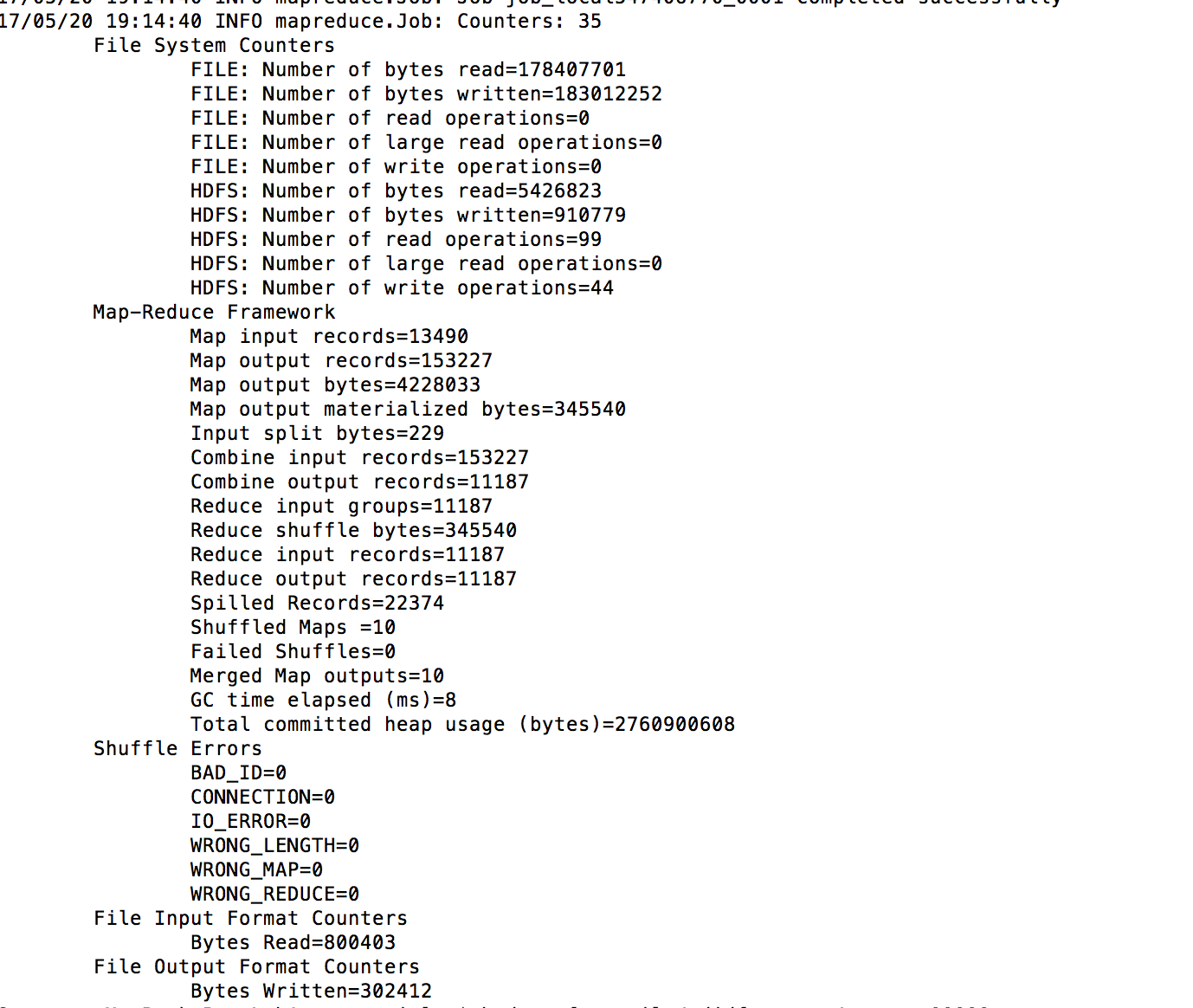
The code for this question can be found in the file **NoaalSDDisplay.java**. The program assumes that the file “isd-history.txt” contains the necessary data, and exists in a directory named “data”, which exists in the same directory as that which the program is being run in. The program first prints a header with the fields “USAF” (for the USAF code), “STATION NAME” (for the name of the station), “CTRY” (for the country id), and “ELEV(M)” (for the elevation in meters). The program writes its output to a file “isd-history-formatted.txt”, and places tab characters for missing fields. A screenshot of the head of the output file is included below:



**Figure 2: Head of Output File from NoaalSDDisplay.java**

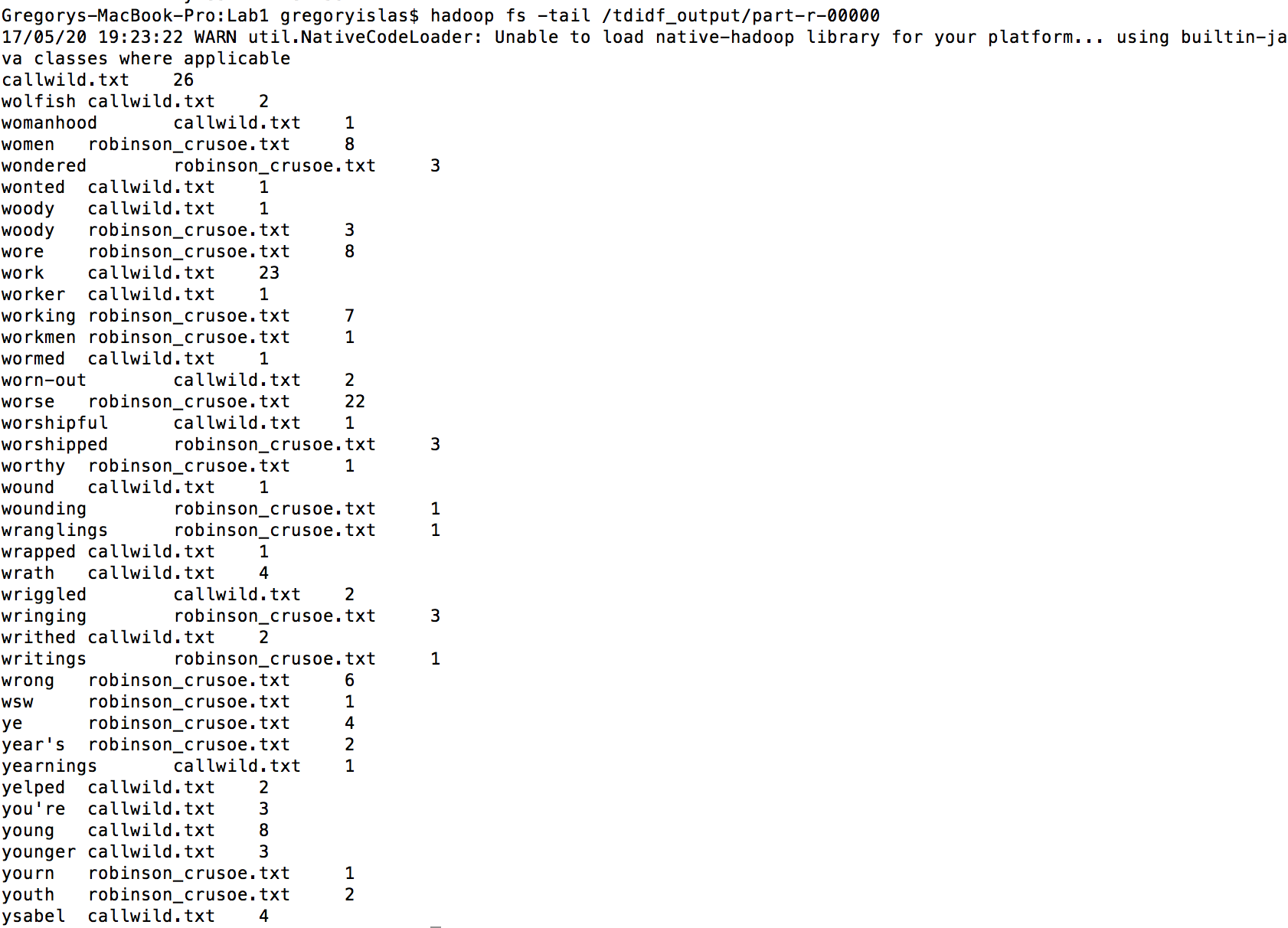
**Question 4.1:**

The process for the TF-IDF calculations take place in three separate map reduce jobs. The code for the first job can be found in the package **wordcount**. This job takes as input the specified directory (in the case for the running of the following jobs, the data used was the two novels *Call of the Wild* by Jack London and *Robinson Crusoe* by William Defoe), and outputs the count for each word by each document. A quick note about what was considered a word: each line of a text file was split by all spaces and double-hyphens, and each token was changed to all lowercase letters. All non-alphanumeric characters were removed aside from hyphens and apostrophes that were contained within tokens/words. The output from the terminal window can be seen below:



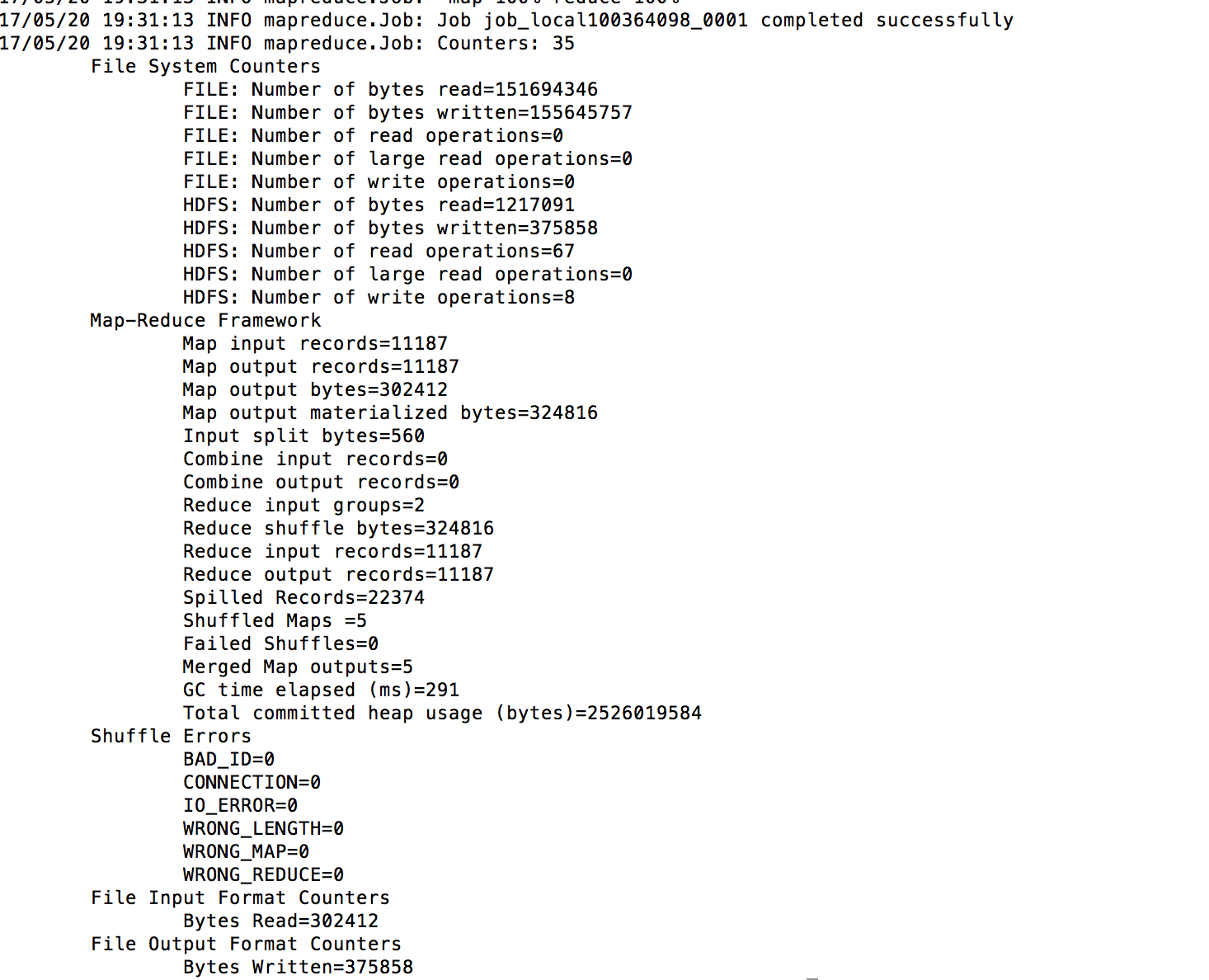
**Figure 3: Output from First Job of TF-IDF (wordcount)**

The tail of the output from the first reducer can be seen in the below screenshot:

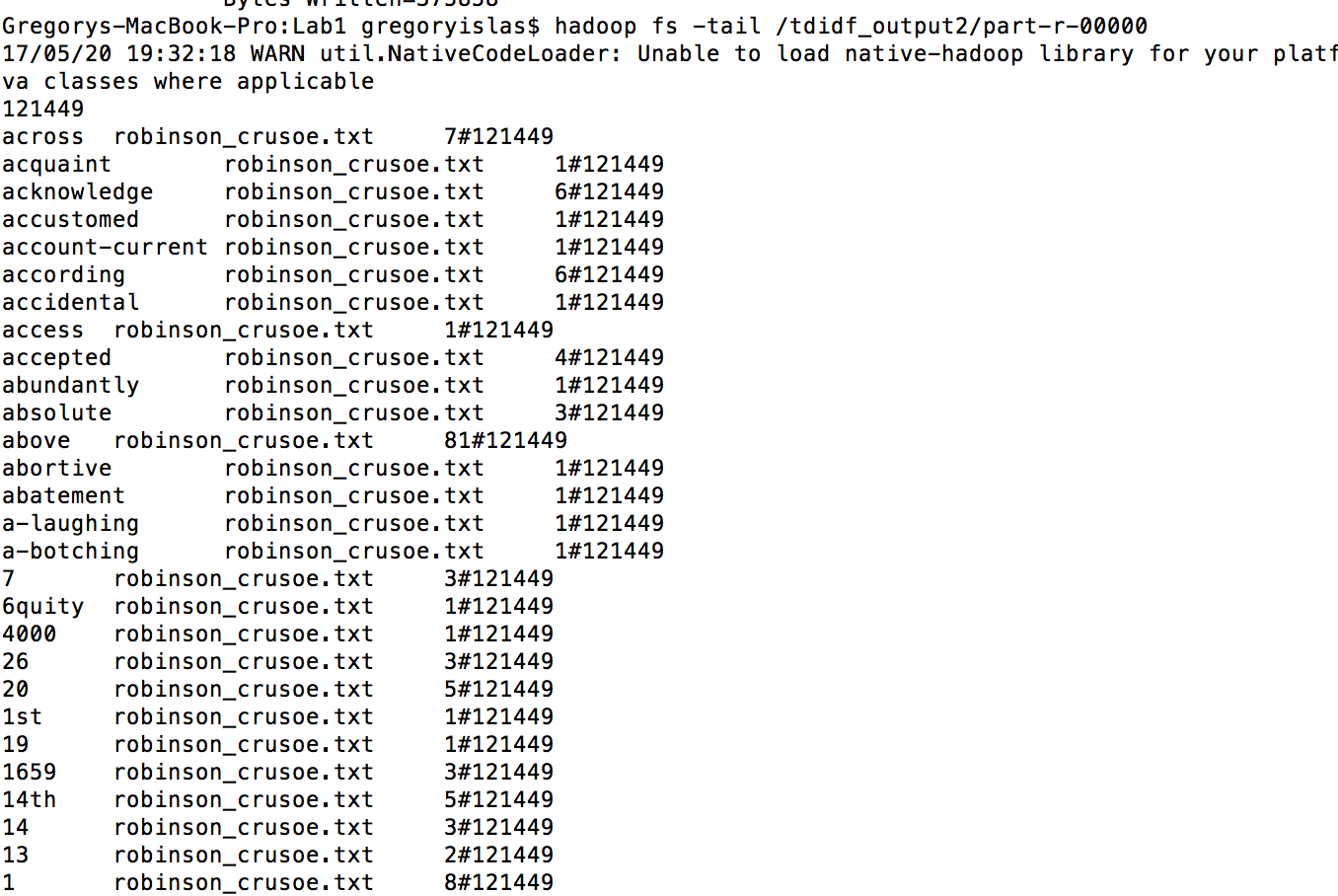


**Figure 4: Tail of First Reducer for First Map Reduce Job of TF-IDF**

The second map reduce job took in as input the output of the first map reduce job, and can be found in the package **wordcount.per.doc**. This job counted the total number of words per each document (it only used one reducer since there were only two total files, and each document name would be used as a key). Figure 5 shows the output logs from this job, whereas Figure 6 shows the result of the output from the tail of the output file.

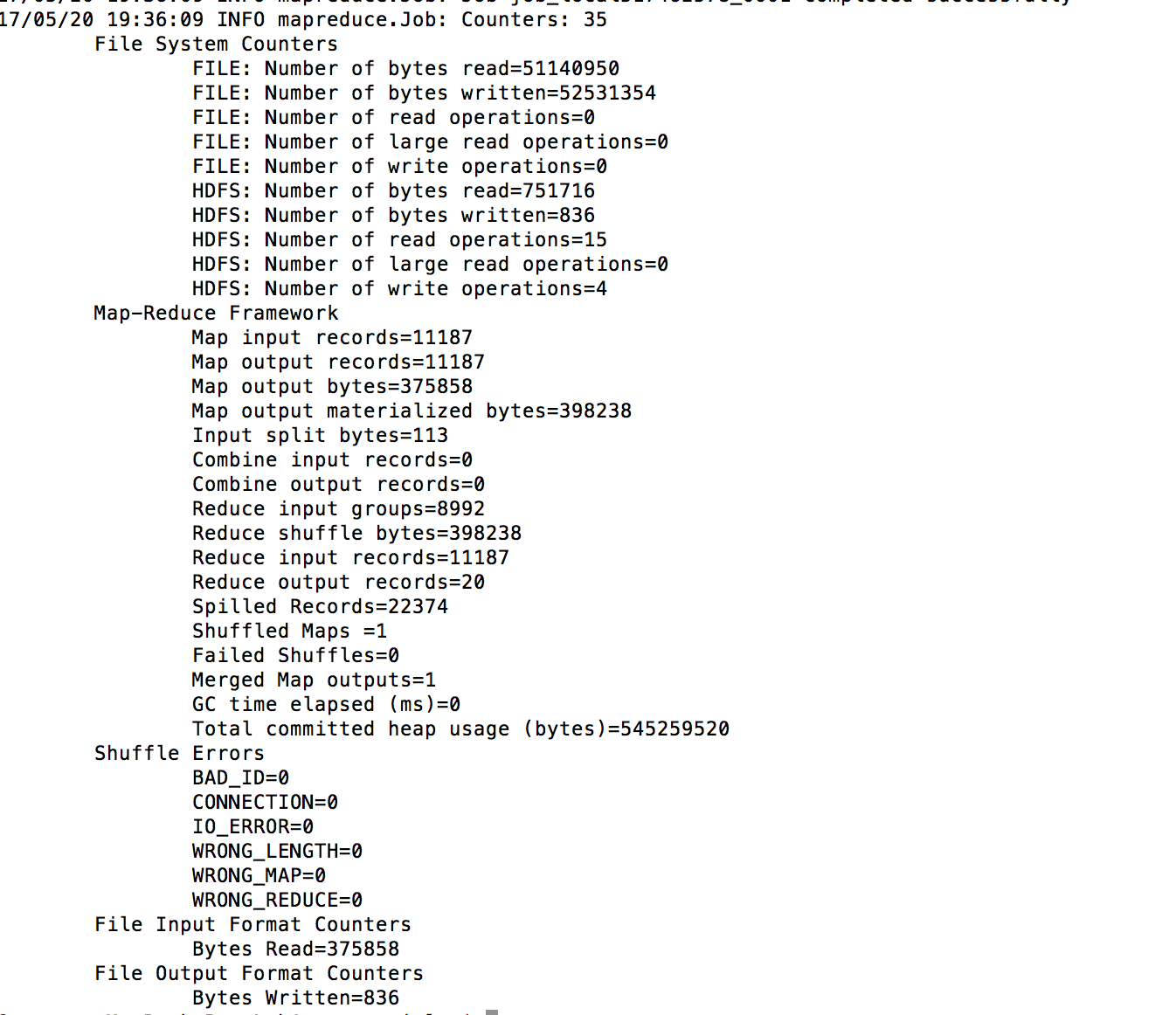


**Figure 5: Output Logs From Second TF-IDF Map Reduce Job**

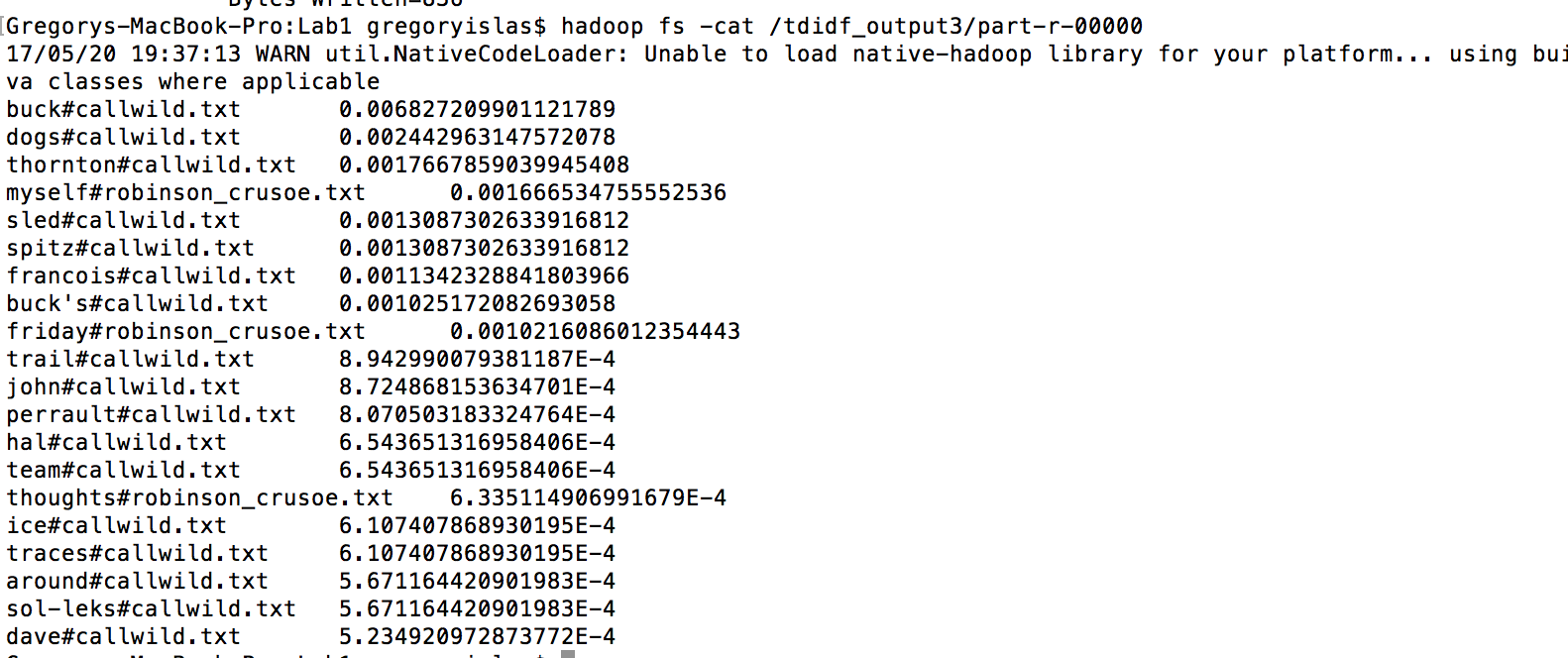


**Figure 6: Tail of Output from Second Round of TF-IDF**

Finally, the last portion of the job can be found in the package **tf\_idf**, which computes the final TF-IDF scores of each word/document pair. This program was designed to only print out the top 20 scores, so it only uses one reducer. However, on larger data sizes, it may be more appropriate to use more reducers, and perform the top 20 sorting on the output of this job. Figure 7 shows the logs from the running of this job, whereas Figure 8 shows the top 20 TF-IDF scores.



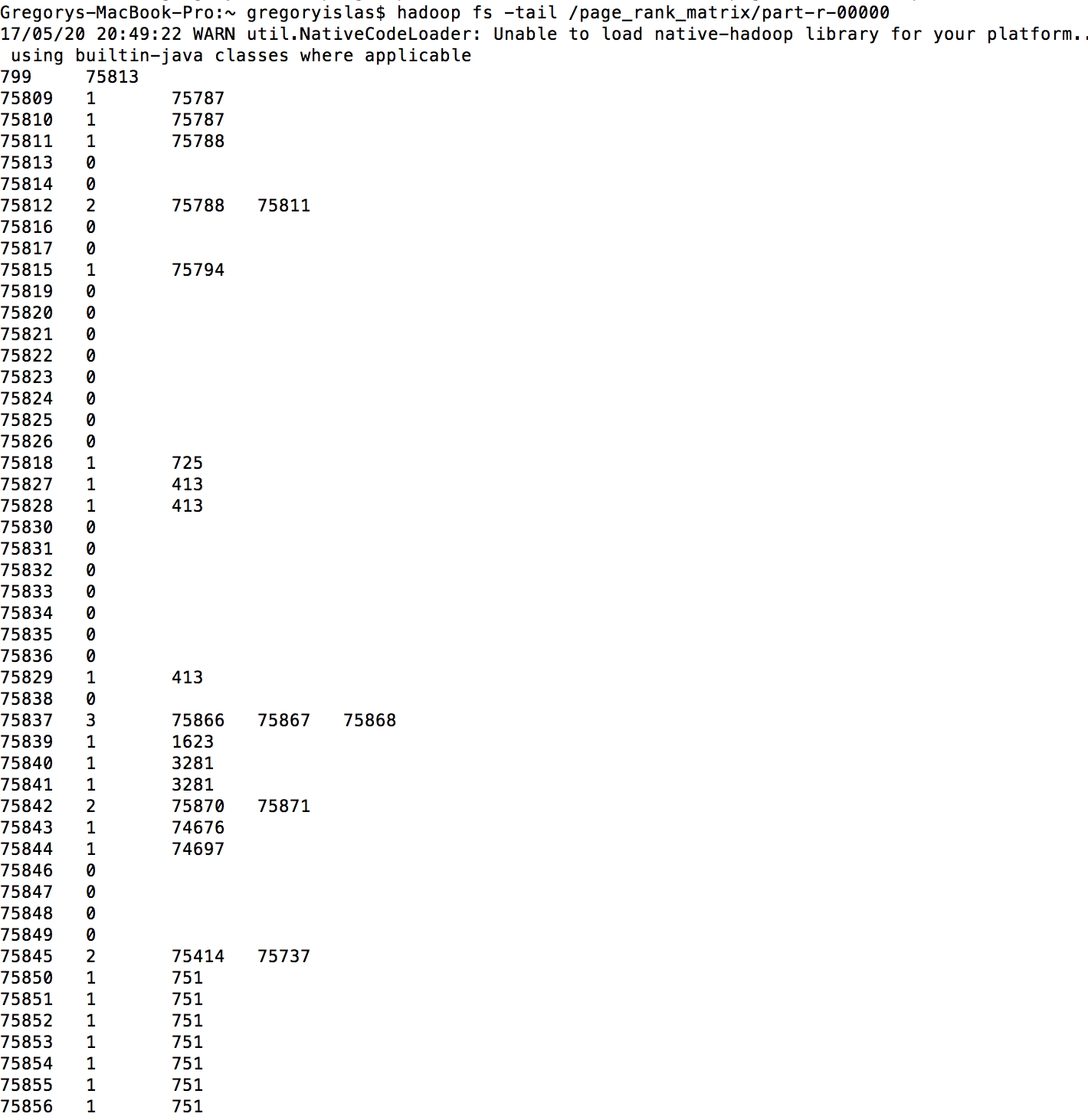
**Figure 7: Output Logs From Final Round of TF-IDF Map Reduce**

****

**Figure 8: Top 20 TF-IDF Scores from Call of the Wild/Robinson Crusoe Corpus**

**Question 4.2:**

To find the top 10 users from the epinions social network, the page rank was computed for each user via the method of repeated multiplication of the page rank vector by the transition matrix until the stationary distribution was obtained. First, the matrix had to be created from the raw text file **soc-Epinions1.txt**. The map reduce job to perform this can be found in the package **build.matrix**, and produces a text file in a matrix of the form described in Chapter 5 of Jeff Ullman’s book. In the file, the matrix is stored with three fields. The first contains the source node, the second contains the number of its destination nodes/edges, and the third contains all the nodes that the source node has edges to. A screenshot of the head of the matrix is shown in Figure 9. It was assumed that there existed users numbered from the minimum value to the maximum value of the user ids.



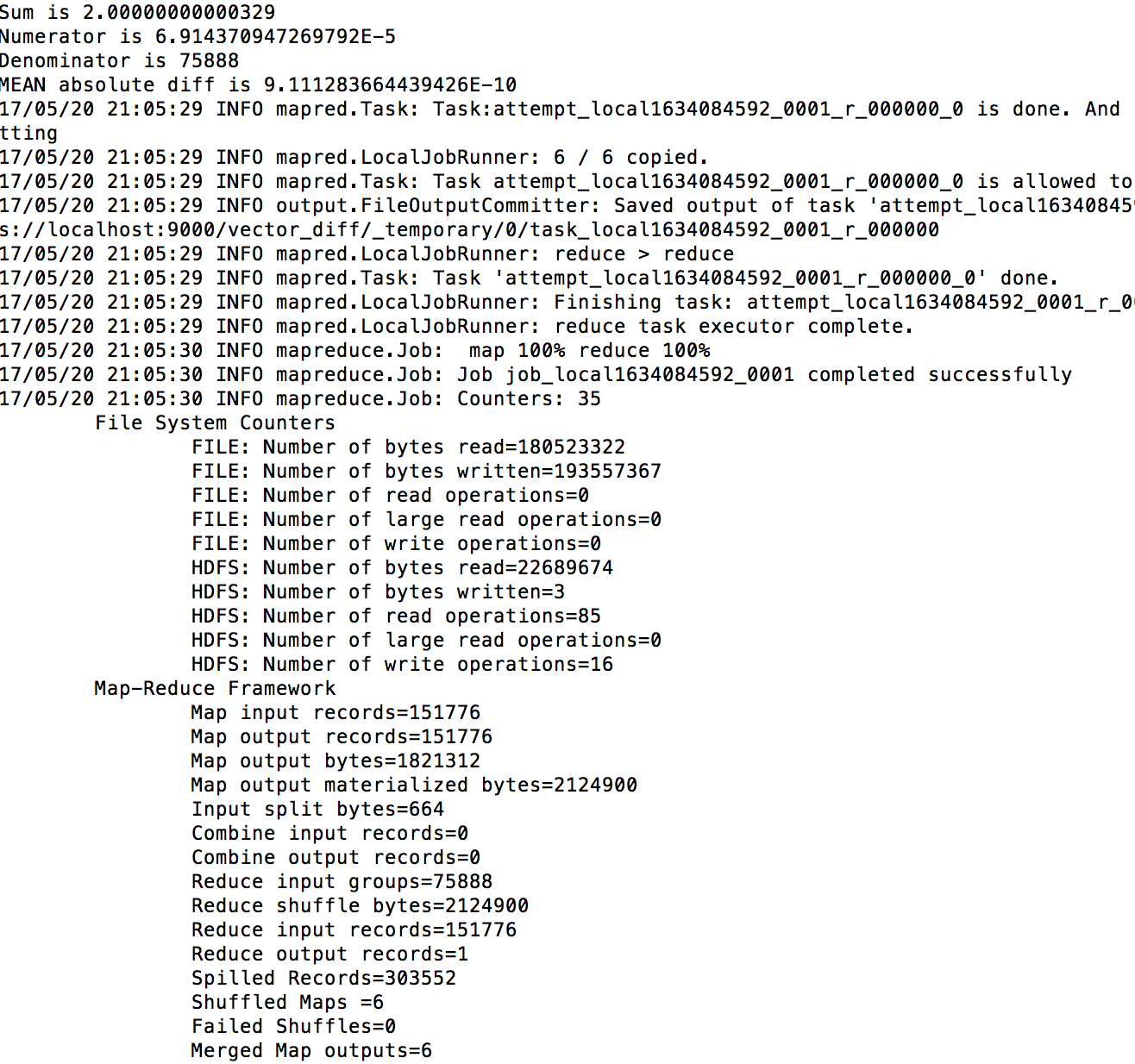
**Figure 9: Tail of Transition Matrix File**

Aside: Since the entirety of the jobs required many rounds of matrix multiplication, a bash script was written to automate the multiplication of the vector/matrix until convergence was reached. Convergence was defined as occurring when the mean absolute difference of the output vector and the input vector was less than 1E-9. The bash script can be found in the file **page\_rank\_script.sh**, though it will likely have to be modified since the directories used were on my local computer.

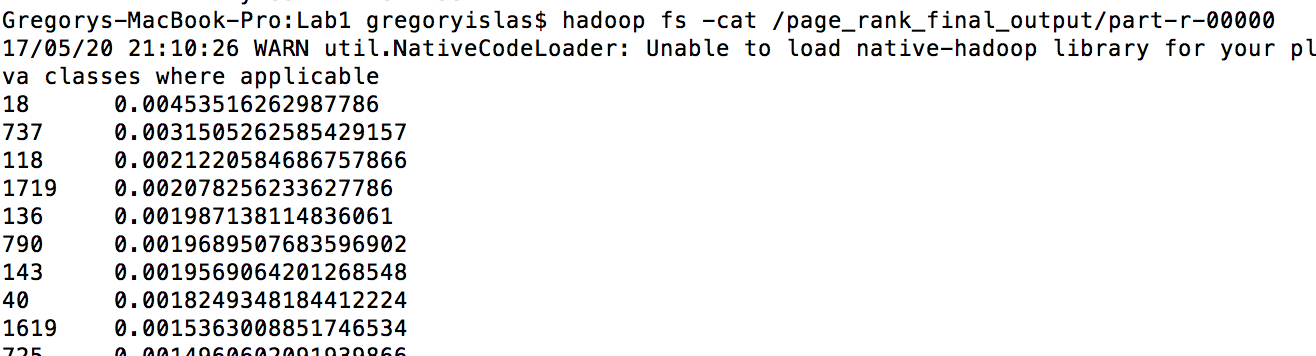
The package **build.vector** contains the code for building the initial vector, which was initialized as 1/N, where N represented the number of users.

The package **multiply.matrix** contains the map reduce job for multiplying the vector and matrix. This job relies on the key assumption that the page rank vector can fit in main memory. If this was not the case, the Map function found in **MapMultiplyMatrix.java** would have to be modified somewhat, or the vector file and matrix files could be split up into blocks for parallel processing.

Once a round of matrix multiplication was completed, the job for comparing the difference between the two vectors can be found in the package **check.vector**. Finally, a very simple job for printing out the top 10 users can be found in the package **output.top10**. To converge, the job took 27 iterations. Figure 10 shows the final output from the check vector job, whereas Figure 11 shows the top 10 users in terms of page rank.



**Figure 10: Final Job Logs From Last Check Vector Round**



**Figure 11: Top 10 Users in Terms of Page Rank**