Big Data Algorithms Techniques and Platforms

Assignment 1

Olivier Salaün

20th May 2017

1 Question 4.1 - TF-IDF

For solving that question, we used three map-reduce that rely on the following files :

- TF_IDF_1.java takes as input the text files in folder input1 and makes an output in folder output1_input2. It takes all the raw text, normalizes it (removes punctuation and set all words in lower-case), and produces a countword list where key is "word ; doc_id" and value is the number of occurrences of a word in doc_id.

- TF_IDF_2.java takes as input the files in folder output1_input2 and makes an output in folder output2_input3. From the previous output, it generates an intermediate file with a new arrangement: "doc_id (tab) word =>wordcount" . Next, in the reducer, for each key doc_id, the code returns as key "word ; doc_id" and as value "number of occurrences of the word in doc_id ; total number of words in doc_id".

- TF_IDF_3.java takes as input the files in folder output2_input3 and makes an output in folder output3. The mapper separates the four elements of each line of previous output and makes an intermediary result whose key is "word" and whose value is "doc_id ; frequency of word in doc_id : total number of words in doc_id". If a word is present in several texts, the mapper will return several lines with the same word as key. In the reducer, we extract the total number of documents contained in folder input1. We use a first for-loop over the values for counting the number of documents in which the word (key) is present and we store the different components of the values within a hashmap. In a second for-loop iterating over the same hashmap whose key is doc_id, we compute, for each doc_id, the term frequency (frequency of the word / total number of words in doc_id) and the inverse document frequency (log of (total number of documents / number of documents in which the word is present)). The product of both values returns TF-IDF. The final output of the reducer returns as key " key ; doc_id in which the word is present" and as value the TF-IDF score.

We have tried to sort directly the TF-IDF scores within the map-reduce without satisfying result. Therefore, we sorted them in an excel file and gathered

Table 1: Top 20 words with highest TF-IDF scores

Rank	Word	Document ID	TF-IDF score
1	buck	callwild	0.006655988
2	dogs	callwild	0.00218168491080484
3	thornton	callwild	0.00188586322798384
4	myself	defoe-robinson-103.txt	0.00143167654644133
5	spitz	callwild	0.00120177558646029
6	sled	callwild	0.00116479787610767
7	francois	callwild	0.00110933131057873
8	friday	defoe-robinson-103.txt	0.000907055346204268
9	trail	callwild	0.000758043062228801
10	john	callwild	0.000739554207052488
11	perrault	callwild	0.000721065351876176
12	hal	callwild	0.000684087641523552
13	team	callwild	0.000628621075994615
14	thoughts	defoe-robinson-103.txt	0.000544233207722561
15	sol	callwild	0.000536176800113054
16	ice	callwild	0.000517687944936742
17	leks	callwild	0.000517687944936742
18	traces	callwild	0.000517687944936742
19	around	callwild	0.000480710234584117
20	dave	callwild	0.000462221379407805

in Table 1 the top 20 words with the highest TF-IDF scores in both documents defoe-robinson-103.txt and callwild.

2 Question 4.2 - Page Rank

To perform Page Rank, we used two map-reduce that rely on the following files:

- page_rank1.java takes as input the text file in folder PR_input1 and makes an output in folder PR_output1_input2. The mapper sets key as departure node and value as arrival node. The reducer makes the following output: key = "departure node", value = "initial PR score set at 1 (tab) arrival node(s)".

- page_rank2.java takes as input the text file in folder PR_output1_input2 and makes an output in folder PR_output2. The mapper returns two types of intermediary outputs for each key: one contains key = "departure node", value = "initial PR score set at 1 (tab) number of arrival nodes" ; the other output contains key = "departure node", value = "*" + "list of arrival nodes". In the reducer, the second output with "*" at the beginning of value is used for storing all destination nodes corresponding to the key/departure node into a string. That string may be used later for further iterative computations of page rank,

Table 2: Top 10 users with highest PageRank scores

Rank	**User**	**PageRank score**
1	18	312.7855523
2	4415	143.997038
3	737	133.272699
4	790	115.8860223
5	1753	114.4899719
6	143	113.9788559
7	1719	113.4423779
8	136	99.60254122
9	751	99.21994567
10	118	85.9018146

but we could not perform them. The first mapper output is used for computing the ratio of initial page rank score (1) over the total number of arrival nodes for each key/departure node. This ratio is used for computing the page rank score that is finally included in the reducer pair-value output. We sorted the scores within an excel file and showed the top 10 users with the highest PageRank scores in Table 2.

References

Jimmy Lin and Michael Schatz. 2010. Design patterns for efficient graph algorithms in MapReduce. In Proceedings of the Eighth Workshop on Mining and Learning with Graphs (MLG '10). ACM, New York, NY, USA, 78-85. DOI=http://dx.doi.org/10.1145/1830252.1830263

Leskovec, J., Rajaraman, A. and Ullman, J. (2014). Mining of massive datasets. 1st ed. Cambridge: Cambridge Univ. Press.

Lin, J. and Dyer, C. (2010). Data-intensive text processing with MapReduce. 1st ed. Milton Keynes: Morgan Claypool.

All computations were made on Cloudera Virtual Machine with CentOS realease 6.7 (Final).