DISTRIBUTED COMPUTING AND STORAGE ARCHITECTURE

STUDENT NAME – TRIPAT KAUR

STUDENT ID – 0572962

Contents

TASK 1 – TOP 10 KEYWORDS FOR EACH MOVIE GENRE

The task required to find the top 10 keywords for each movie genre.

Each line of the file is read by the mapper (function name: mapper_get_words). In the mapper pre-processing is performed. This includes the following:

1. Removing the year when the movie was released and all other numbers from the title of the film.
2. Removing all special characters except alphabets
3. Removing all stop words from the movie title. Stop words such as ‘a’, ‘the’, ‘me’,’too’ etc.
4. Removing all extra spaces generated by the previous three steps.

Since a movie can belong to multiple genres, the genres are split into lists.

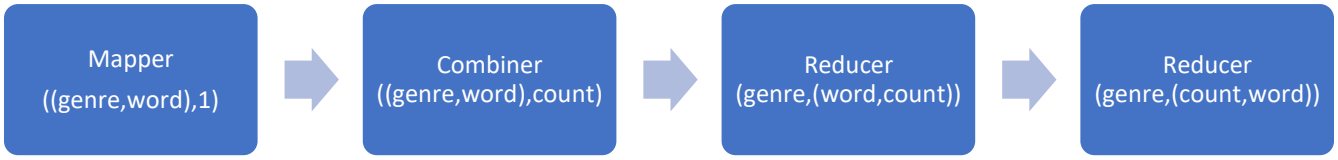

Figure 1 Map Reduce Task 1

For each movie the mapper yields ((genre,word),1).

The output of the mapper is sent to the combiner (function name: combiner_count_words) which sums up all genres and word pairs.

The output of the combiner is then sent to a reducer (function name: reducer_count_words) which splits the ((genre,word),count) pairs to (genre,(count,word)) so that all common genres and can be sent to the same reducer.

The last reducer (function name: reducer_find_max_word) takes in the genre and sorts the word list by the count in descending order and the top 10 words are returned along with their count of occurrence.

```
Streaming final output from C:\Users\HP\AppData\Local\Temp\Task1_Top10WordsBYMovieGenre.HP.20211128.095355.286396\output...
"(no genres listed)"    [75, "la"]
"(no genres listed)"    [68, "love"]
"(no genres listed)"    [60, "man"]
"(no genres listed)"    [55, "story"]
"(no genres listed)"    [52, "de"]
"(no genres listed)"    [40, "life"]
"(no genres listed)"    [34, "live"]
"(no genres listed)"    [34, "le"]
"(no genres listed)"    [32, "last"]
"(no genres listed)"    [30, "one"]
"Action"        [117, "man"]
"Action"        [95, "ii"]
"Action"        [77, "death"]
"Action"        [72, "black"]
"Action"        [64, "last"]
"Action"        [62, "vs."]
"Action"        [62, "dragon"]
"Action"        [61, "blood"]
"Action"        [60, "movie"]
"Action"        [60, "dead"]
"Adventure"     [64, "movie"]
"Adventure"     [60, "man"]
"Adventure"     [57, "adventures"]
"Adventure"     [55, "king"]
"Adventure"     [55, "island"]
"Adventure"     [54, "lost"]
"Adventure"     [52, "last"]
"Adventure"     [49, "world"]
"Adventure"     [48, "dragon"]
```

Figure 2 Task 1 Result Screenshot

TASK 2 – REVERSE WEB-LINK GRAPH

The task required to find the reverse web links from a given text file.

Reverse web link means finding all the sources for a particular target i.e. from what all links can one reach to a particular target.

The mapper (function name: mapper_get_target) takes in as input each line of the file and generates the (target,source) pair. This is then sent to the reducer (function name: reducer_target_list_source) which combines all sources of the target to a list.

Figure 3 Map Reduce Task 2

```
215089  [213297, 911963]
21509   [229103, 132693, 412304]
215090  [618953]
215091  [415356]
215092  [21640]
215093  [614978, 690532, 738140, 797248, 805113, 439436, 617503, 691586, 739146, 747717, 781047, 339614, 680829, 706224]
215095  [869130, 156337, 243978]
215096  [189196, 692481]
215097  [492007, 20910, 131658, 508542, 523025, 758803, 71287, 415869, 130744, 713604, 883849, 152767, 206192]
215098  [408408]
215099  [544296]
2151    [832058]
21510   [18659, 700340, 122987, 793399, 171064, 355811, 423044, 628016]
215100  [908696]
215101  [271170, 515863, 550726]
215102  [826638]
215104  [75998]
215106  [438430]
215107  [93347]
21511   [264984, 740772, 760852, 871520, 355406]
215111  [870588, 124405, 720903, 635200, 829285, 834840]
215112  [401980, 59844, 371487, 642153, 706855, 205528, 93886, 164649, 311486, 354416, 376558, 428683, 695624, 731418, 804692, 903290, 797135, 313
231, 441571]
215113  [152028, 230774, 236672, 320058, 548621, 559645, 681958, 892276, 839290]
215114  [380747, 135998, 175093, 216827, 533227, 805809]
215115  [230144]
```

Figure 4 Task 2 Result Screenshot

TASK 3 – K NEAREST NEIGHBOR

The task 3 requires the implementation of k-nearest neighbors in a distributed fashion.

Since a lot of pre-processing is required such as normalization of the data, the file is read outside the mapper function first and the training data (i.e. rows of the file where labels are available) are stored in an attribute of the class. The training data is then normalized using MinMaxScaler from the scikit library.

The mapper (function name: mapper) takes as input every line of the file but processes only those where the labels are missing. For each test record, the test record is first normalized and then the euclidian distance from every training point has been calculated. For each training point, the euclidian distance and the label are stored in a list of dictionary. The lists are sorted based on the value of the euclidian distance. The mapper generates the id of the test record and the top (k=)15 (as per requirement) neighbours.

The ouptput of the mapper is then sent to a reducer (function name: reducer) which counts the number of occourences of the labels for top 15 neighbours. It yields as output the most commonly occurring label which is the prediction.

Figure 5 Map Reduce Task 3

```
No configs found; falling back on auto-configuration
No configs specified for inline runner
Creating temp directory C:\Users\HP\AppData\Local\Temp\Task3_KNNClassification.HP.20211128.102613.744857
Running step 1 of 1...
job output is in C:\Users\HP\AppData\Local\Temp\Task3_KNNClassification.HP.20211128.102613.744857\output
Streaming final output from C:\Users\HP\AppData\Local\Temp\Task3_KNNClassification.HP.20211128.102613.744857\output...
"39"    "Iris-versicolor"
"54"    "Iris-virginica"
"96"    "Iris-setosa"
```

Figure 6 Task 3 Result Screenshot

TASK 4 – FROBENIUS NORM OF A GIVEN MATRIX

Frobenius Norm of a matrix is the square root of the sum of the squares of the elements of a matrix.

The task required to calculate the frobenius norm of a matrix given in a text input file.

The mapper (function name: mapper_get_row) splits the line of every input to generate a list of values. It returns a unique id of the row and a list of elements belonging to that row.

The reducer (function name: reducer_row_intermediate) takes in all elements of a row and calculates the sum of square of elements. It generates a null key and square value as its output. The key is kept as null because output of all reducers should go to a common reducer in the last step.

The final reducer (function name: reducer_final) takes in the output of all reducers and sums them up and then calculates the squareroot to generate the frobenius norm of the matrix.

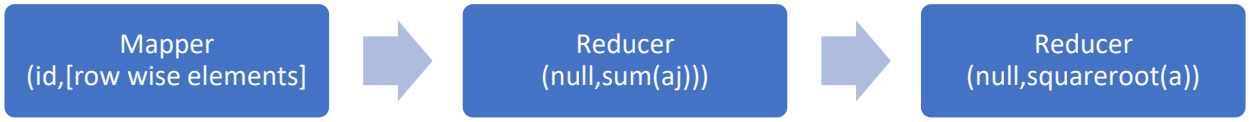

Figure 7 Task 4 Map Reduce

```
Running step 1 of 2...
Running step 2 of 2...
job output is in C:\Users\HP\AppData\Local\Temp\Task4_FrobniusNormMatrix.HP.20211128.110109.163414\output
Streaming final output from C:\Users\HP\AppData\Local\Temp\Task4_FrobniusNormMatrix.HP.20211128.110109.163414\output...
null    32.13209859958961
```

Figure 8 Task 4 Result Screenshot