CSE-587 Assignment 2

SAI CHARITH REDDY PASULA	SANKEEERTH TELLA	PRUDHVEER REDDY KANKAR
Sai chari	stella3	prudhvee
50320452	50317364	50320121

Task 1

In this task, we have to find the number of times that a word appears in the documents. THe input documents are named arthur.txt,

Mapper:

Mapper reads lines from input files using sys.stdin, the lines are striped and converted to lowercase and the words are split using space
To find the word count, we first have to preprocess the documents to remove punctuation marks from the data. This is necessary because suppose we have words like true? And true. These both are the same words but they will have different counts due to the different punctuation marks. To get over this, we used a list of punctuations and we checked each word to see if it has a punctuation and if found, the punctuation is replaced by an empty string. After this the mapper prints each word with a number of 1 separated by tab space.

Reducer:

The reducer takes the words sorted in alphabetical order with count of 1 and combines similar words while adding to the counts of each word.
Finally we get the word with corresponding count. To do this, we use word and current word. As long as the word is equal to the corresponding words, the count is increased and once the current word!= word, the current word is changed to the next word and so on.

Task 2

In this task, we find the top 10 trigrams present in the three documents using the three keywords: Science,sea,fire.

Mapper1:

In the first mapper, we remove the punctuations similar to the first task and also we remove the stop words. To remove stopwords, we use nltk.stopwords.

We replace the three keywords science as *1, fire as *2 and sea as *3. If we find a trigram in which there are 2 keywords, We are manually converting those trigrams into two seperate trigrams. To do this, we check if the number of * in a trigram are 2 if they are we replace each * with $ in separate steps to get multiple trigrams.

Reducer2:

We have 8 reducers in this task. Each reducer outputs 8 different top 10 trigrams. In the reducer file,we put the trigrams into a heap data structure from which we get the top 10 trigrams.

Mapper2:

The second mapper just recieves the 80 trigrams and passes them on to the next reducer.

Reducer2:

In the final reducer, the top 10 trigrams from the top 80 are retrieved using the heap data structure.

Task 3

In this task we have to implement a MapReduce algorithm to produce inverted indexes for the given dataset.

Mapper:

The mapper reads all the lines inputted using the stdin. Then we strip the lines and use the findall function to iterate over all the words in the given document. Similarly we also store the doc_id for each word which is the document to which the word belongs. Then we send both the word along with the doc_id separated by tab space to the reducer.

Reducer:

In the reducer we take each individual line as input and then assign a default list to each word present in the line to which the postings will be saved. Then we iterate through each word and append the postings to the list assigned. FInally we output the word along with their postings list which will be joined using comma.

Task 4

In this task, we have to implement a MapReduce algorithm which can join two datasets using a primary key which in our case is the Employee ID. We convert the input files into .csv files using tab separator. We have attached the csv files that we have used along with the python files.

Mapper:

The xlsx file which was given is converted into a csv file using delimiter as a tab. Since the number of columns in each dataset is different, we compare the number of columns in each dataset to assign then values their respective fields (emp_id, name, salary, country, passcode). The empty fields are set to NA and sent to the reducer.

Reducer:

In the reducer, we take each line from the mapper as input. We create two new dicts for two datasets. In the first dataset, there will be only employeeId and name, we recognise this and add the name to respective employee ID in join1_dict. Else we add employee id, salary, country and passcode in join2_dict. Finally we take each employee id as primary key and comparing it with join1_dict and join2_dict and output name from join1_dict and rest of the fields from join2_dict.