Project -2 Implementation.

Abhishek Madav #86378148

Map/Reduce 1

The first series of the mapper and reducers within the GetFrequency.java implements a basic word count for each token in the input corpus with each key to be a string of token and the location. For each mapper in the system, the output of the mapper is of the form of <token location, one>. The reducers aggregate the list of the mapper output to sum the frequency of each word for a specific location. The output of the reducer stage is series of <token location, frequency>

Map/Reduce 2

The second series of the mapper and reducers within the ProbabilityCalculator.java implements logic to calculate the required probability. The mapper for this stage clubs the token and their frequency against the position of the token. The output of the mapper phase can be a series of key/value pairs of the form <location, token frequency>. The reducer phase aggregates all such locations and calculates the probability using the idea that while calculating the frequency of a token t, number of sentences N be the length of the values pairs with at least t number of token in it. The reducer emits <token location, probability>

Map/Reduce 3

The third phase to get the top three sentences with the highest probability implements mapper to send the values of the <token location, probabilities> from the reducer 2 output to the reducer 3 as a list of values. The reducer implements a distributed cache wherein the corpus is read in the configure function to populate a sentence bucket. A hashtable stores the probability values for the token and its location and each sentence is calculated for its probability as a product of each tokens' probability. An array of three, stores the top three sentences and outputs the same in <sentence, probability>