Description

A map-reduce program that uses Google 3-gram corpus in order to calculate the probability for a word to appear after 2 given words. The program is a Hadoop program, designed to run on AWS EMR & S3.

The probability is calculated with the following formula:

$$P_{del}(w_1...w_n) = \frac{T_r^{01} + T_r^{10}}{N(N_r^0 + N_r^1)}, \text{where } C(w_1...w_n) = r$$

Where:

- N is the number of n-gram instances in the whole corpus.
- N_r^0 is the number of n-gram types occurring *r* times in the first part of the corpus.
- T_r^{01} is the total number the n-grams of the first part (of N_r^0) appear the second part of the corpus (instances).
- N_r^1 is the number of n-gram types occurring *r* times in the second part of the corpus.
- T_r^{10} is the total number the n-grams of the second part (of N_r^1) appear in the first part of the corpus (instance).

HadoopLocal: its purpose is to upload the 4 steps' jars, and download the final result to the local computer.

Step 1 – CalcDataForFormula: a map-reduce program. The mapper filters-out stop-words and invalid inputs, then writes to Context the 3 words as key with the value 1. The reducer separates the given pairs into two sub-corpuses as in the formula and calculates N, N_0, N_1, T_1, T_2. Once the reducer done, it writes the calculated values and tables to context.

Step 2 – CalcProbabilityPerOccurrence: a map-reduce program. The mapper split the given value and extracts a key, which is the name of the hash-map (N\\$N_0$\\$N_1$\\$T_1$\\$T_2$) and the key (the 3-gram). The mapper also extracts the value – the number of occurrences for the 3-gram. The reducer collects this key-values to final hash-maps. Once finished, the reducer calculates the probability of each r value (number of occurrences) according to the formula, and writes the pair <r, probability> to context.

Step 3 – MatchProbabilityTo3Gram: a map-reduce program, similar to classic word-count program with few adjustments. The mapper takes 3-gram as an input and writes to context the 3-gram as key and 1 as value. The reducer downloads to its memory the probability table that was calculated in *Step 2*, sums the occurrences as in word-count and matches the summed value to a probability in the table. If not found (due to the fact that the given formula can cause problems), the reducer will give this 3-gram the probability of the closest number of occurrences in the table.

Step 4 – SortProbabilityTable: a map-reduce program, using the method of secondary sort. The program sorts the table in lexicographic order for the two first words in the 3-grams, and those who has similar 2 first words are sorted by their probability, decreasing. The mapper makes a key out of the 3-gram and the probability, the value is the probability. The reducer deletes the probability from the key and write <3-gram, probability> pair. The new sorting class is NewOrder.

Manual

To run the program, please make sure that the following are in the same directory as *HadoopLocal*:

1) Step1.jar
2) Step2.jar
3) Step3.jar
4) Step4.jar

The output will appear in this directory as a file named "triGramProbability.txt".