Big data management

Assignment 3 : Jaccard distance

Marin Julien

October 9, 2018

1 Result analyze

Gram size	Result size	Running Time 1VM	Running time Multiple VM
2 (bi-gram)	4772	141s	130s on 2VM
3 (tri-gram)	3582	129s	110s on 2VM

2 Description of the solution

The solution uses two map reduce processes. The first one generates the potential pairs, and the selection the pairs if their jaccard distance are between 0.85 and inferior to 1.

The first map reduce process works as follow : It uses a map-side join with the distributed cache. We justify this as the data used has little size (10k lines). The output from the mapper is <Word1, Word2> where word1 comes from the first set, and Word2 comes from the second one. The reducer does not do anything here, as it is not needed for any operation that our data is centralized.

To reduce the tremendous quantity of data that was produced, optimization had to be done. In order to optimize with little cost operations, words' size has been used : Words whose size are very different can for sure not be at least 85% alike, therefore they should be filtered to avoid further useless computation. Let us determined the needed length of a word$_1$ according to another word$_2$ in order to allow its ngrams to be 85% alike. Let len$_{\text{grams}}$(Word, n) be the number of n-grams for a given word.

$$len_{grams}(Word, n) = len(Word) - n + 1$$

We want that word1 and word2 share from 85% to 100% common ngrams, and word1 might be bigger or smaller. We therefore used the following formula :

$$len_{grams}(word_2, n) * 0.85 \leq len_{grams}(word_1, n) \leq \frac{1}{0.85} * len_{grams}(word_2, n)$$

$$= (len(word_2) - n + 1) * 0.85 \leq len(word_1) - n + 1 \leq \frac{1}{0.85} * (len(word_2) - n + 1)$$

Only pairs of words fulfilling this condition will be outputted by the map.

During the second mapreduce phase, all the grams are generated, and the jaccard score is computed out of it. For each pair, the ngrams are generated and the score $\frac{|ngrams_1 \cap ngrams_2|}{|ngrams_1 \cup ngrams_2|}$ where ngrams$_1$ and ngrams$_2$ come from respectively word1 and 2. All this process is done throughout the map to enjoy its scalability once more. Its output is <Word1, Word2>, selecting the pairs of words whose score is superior to 0.85 but not 1. The second phase is probably still optimizable.