Project 4-Set Similarity Join Using Spark on AWS

COMP9313 Big Data Management

Tianwei Zhu, z5140081

A simplest way to do this project is to use two circulations to compare each line with the rest lines. But this method is time consuming and is not suitable for big data.

First of all, I calculate the prefix length for each line by $|line| - \lceil |line| * threshold \rceil + 1$. Then map each line to number of prefix lines, which are formed by token as key and original line as value (tokens of prefix # front of the line). This function is called conbine_frequency and it is the preprocessing work. (I will talk about how to sort and pick by frequency later).
After this, we can get a RDD in form of Set[(Int, Array[Int])]. The front int represents tokens and last one is an array storing lines. I used flatMap to unfold RDD to (Int, Array[Int]) into several key-value pairs.

Now we can compare each pair from previous RDD, but we still need to do some pruning.

1. I compare each line with the posterior lines to avoid repeated comparisons.
2. I set a container (I used "set" because it is faster) to store all index pair which were compared before. If there are two lines been compared before, just skip this turn. This can make program much faster.
3. We should notice that for two list ζ and ζ', we can have:

$$\text{Jaccard}(\ell, \ell') = \frac{|\ell \cap \ell'|}{|\ell \cup \ell'|} \geq t \Rightarrow \frac{|\ell'|}{|\ell|} \geq t \Rightarrow |\ell'| \geq t \cdot |\ell|$$

   This means we can only consider the two lists which satisfied this condition. So another condition has been added in my program: if((words_2.length-1) >= thres*(words_1.length-1)
4. There are also some small optimizing, such as using Int instead of String, broadcast the frequency list, make full use of one RDD but not repeatly creating new one.