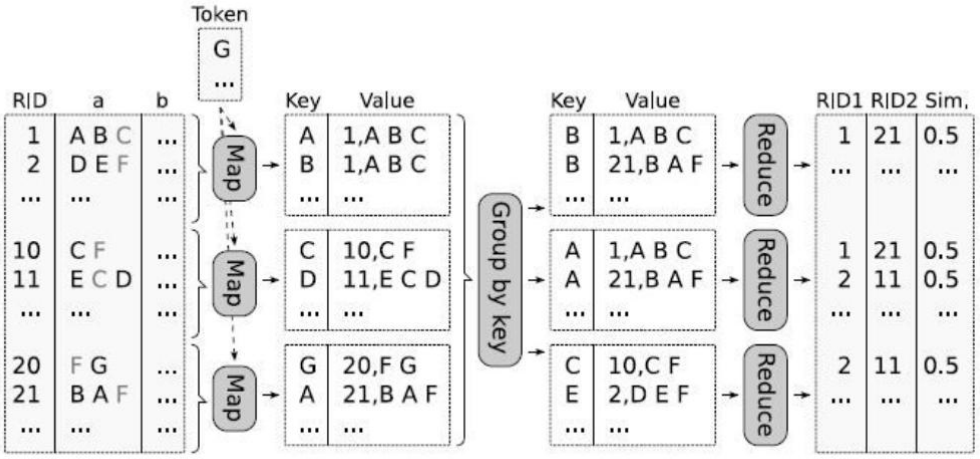
1. At the start of the project, I used 5 min to finished a simple code to solve this problem used
Cartesian(), which will take n^2 complexity, but our goal is to find a efficient way ,so it is definitely not work
2. Then I learned the algorithm for PPjoin, then I trying to implement it with map reduce followed this graph



My code works fine but takes 80min to complete london.txt
Originally it structure is
Split each line into array
.Map each line id,content to content(x),id,content
.Group them by content(x)
.use two for loop to get all the passable pairs for each content
.calculate the similarity for each pair
.filter all the pairs with first number > second number
.filter all the pairs with similarity > t
.map to the correct format

Which has too much repetition, so I mixed filter with all the if statement,
After that my code cost 40 min,

3.
After solve the Useless repetition problem of my code, I'm thinking about to reduce complexity from logic,
I find

Prefix filtering principle:
for text x,y sorted by token,
if O(x,y) >= a, first | x| - a + 1 in x have at least one coincidence with first | y| - a + 1 in y
Which mean for a sorted line X,only need to choose first |x|-ceil(|x|*t) +1

And

Length filtering principle：
If we have two lines
“a”
“a”,”b”,”c”,”d”
The largest similarity for those two lines are 0.25
So If t greater than 0.25,then we dont need to calculate two line with length 1 and 4, which is

for any text x, y, |y| <= t*|x| (|x|>|y|)
We dont need to calculate their similarity,because their largest similarity it less than t

After those optimization, runtime is shortened to 26min,

4.At the last I modified some redundant duplicates in the code again to let in shortened to 23min on VM, and 16min on GCP with 3 worker,It's not the simplest number but I'm happy with it