Report: Homework 1

Finding Similar Items [Simi]

Khaled Jendi

2018-11-11

Introduction

The homework is based on how to find similar text between two documents. It is divided into main four parts, shingling (construction of hashed k-shingles of a document), comparing these documents by comparing the set of shingles using jaccard similarity, computing min hashing and producing signature matrices, comparing two documents by comparing the jaccard similarity of signature matrix.
One more step that is seems to be interesting, is that the calculation of time elapsed (execution time) of both comparing sets and comparing signature matrices.

The project has been implemented in scala as sbt project where no other external framework has been used (standalone application). Note that the project source code has scala docs for better understanding for each functionality of the project. The test dataset is taken from ConvNetWord[1]

Reader and Shingling

The first step was by loading the dataset of documents in the project and having them as list of text where each text represents a document. This is accomplished in Reader class. After the list of documents (as text) has been fetched, the Main method defines an instance of shingling class to run shingling process. It creates collection of sets where each set have all unique shingles for a document. It also maps each shingle of all documents and sort them. Lastly, it generates the boolean matrix of shingles, where 0 means this shingle is not represented in the document, and 1 means the shingle is represented in the document.

```
Boolean matrix (shingles of documents)
10001011000100000110100110
11011111111111111110101111
10001111100110011110101111
10001111100110011110101111
11011100100101000000000111
11011100100101000000000011
11010100100100000000000011
10000100100000000000000010
10000000000000000100000000
10010100000000010100000100
10010100000000010100000100
10110110000000011110000110
11110111101111111111111111
10010110000000010111000110
11110111000001101110001l0
...
Total documents (columns): 25, total shingles (rows): 296955
```

Representation of boolean matrix

Comparing Sets

For comparing sets of shingles, a `CompareSets` class is defined is defined along with `jaccardSimilarity` function which uses the equation: sim(c1, c2) = (c1 ∩ c2) / (c1 U c2) to compute the similarity between two documents. Scala has built-in intersection/ union between two sets or lists.

```scala
/**
  * computes jaccard similarity between two sets
  * @param setA first set
  * @param setB second set
  * @return return similarity percentage
  */
def jaccardSimilarity(setA: Set[Int], setB: Set[Int]): Double = {
  val intersection: Double = (setA intersect setB).size
  val union: Double = (setA union setB).size
  BigDecimal(intersection / union).setScale(2, BigDecimal.RoundingMode.HALF_UP).toDouble
}
```

So, in Main function, the similarity between the first document and all other documents are computed, also the user can select to compute similarity between any of these documents. I noticed when reducing the number of k, the similarity becomes more accurate (such as k=5). For example, the word "editorial" and "factorial" are not similar if k = 10, but they are similar if k=4

MinHashing

For the implementation of min hashing, as mentioned in the lecture and slides. In the primary constructor of the class, define k is defined as k=100 as an independent has functions, and I define list a which is filled with random numbers between 0 and 100. Also, list b which is defined as (2 multiplied by random number between 0 and 100 + 1). These are random functions to support permutation for min hashing.

The `computeSignatures` function is used to generate signature matrices for each set of shingles (document). This is done by taking each column of boolean matrix (which represents a vector of 0's and 1's if shingle represents) and then the signature matrix is defined and filled with max value of integer, so that we can find minimum of hash and assign it later on. Lastly, the loop through all signatures to compute the minimum signature if the value of the column of the matrix array is 1. The minimum signature is calculated by using this equation: h(x) = (ax+b)%c. lastly, if the current value is smaller than what we have in the signature array, then we update the signature array with the new value

Finally, list of signature arrays (signature matrix) is returned.

Compare Signatures

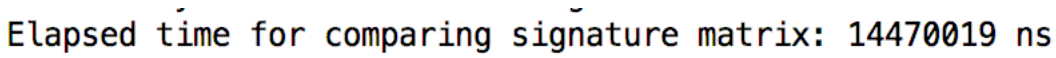
Lastly, to compare documents, now we can compare signature matrix instead of comparing sets. For this, the `CompareSignatures` class is defined. It does exactly the same as compare sets, by comparing jaccard similarity between two signature arrays.

Execution time

A time elapsed function has been used to test the execution time between comparing sets and comparing signatures. When comparing sets, it takes 136617604 ns ≈ 136.66 ms

```
Elapsed time for comparing sets: 136617604 ns
```

While it takes much shorter time when executing comparison of signature matrix 14470019 ns ≈ 14.47 ms.



Run the code

The code is standalone sbt scala project which does not need any external libraries or scripts to run. To run the project, we can import it in intellij, and run Main function.

The dataset is taken from ConvNetWord/data/opi
Project also can be found in: https://github.com/jSchnitzer1/Simi

References

[1]: https://github.com/braveld/ConvNetWord/tree/master/data