ComS 535x:
Project Report #2

Due on Mar 8, 2015

Instructor: Professor Aduri, Pavankumar

Yuanyuan Tang, Chenguang He

Contents

1 MinHash **3**
1 Your procedure to collect all terms of the documents and the data structure used for this. 3
2 Your procedure to assign an integer to each term 3
3 The permutations used, and the process used to generate random permutations 3
4 Brief pseudo code for methods: exactJaccard, minHashSig, approximateJaccard, minHashMatrix . . . 3

2 MinHashAccuracy **6**

MinHash

1 Your procedure to collect all terms of the documents and the data structure used for this.

I use Scanner to scan each file in the given folder, and save all modified terms in $HashSet < String > terms$. Also, i use $HashMap < String, HashSet < String >> allStringsPerFile$ to store the file name and its terms. The reason i pick HashSet to store the term is that, when we construct the binary representation of terms, we can have O(1) access to the terms.

2 Your procedure to assign an integer to each term

I make a iterator, iterate all the terms one by one. I start a counter j at the beginning in iteration. For each term, if it is belong to the file, i put the counter in output array. Therefore, we get a array of a sequence with all terms appeared in increasing order. In here, we use Hashset instead of Treeset is kind of tricky. Since the iterator of Hashset is not in order, it could result in the different iterator returns after we add new element in Hashset. However, after we read files, the Hashset is final. It will not change, that is why i pick up Hashset insteadof Treeset.(Notice that, Treeset's iterator is in order and stable).

3 The permutations used, and the process used to generate random permutations

I pick a very large prime p at the beginning, p ∈[N, 2N], where N is the number of terms. Then i use (a+bx)
To generate p, i use prime test from N. and it randomly comes a number in [N, 2N]. By the prime number theorem, the average number of prime in [N,2N] are $\frac{N}{ln(N)}$, therefore there must exist enough prime numbers in the range.

4 Brief pseudo code for methods: exactJaccard, minHashSig, approximateJaccard, minHashMatrix

Algorithm 1 Exact Jaccard Calculation

procedure EXACTJACCARD($file1, file2$)
  b1 $\leftarrow$ $termsBinaryRepresenatation$ of file1
  b2 $\leftarrow$ $termsBinaryRepresenatation$ of file2
  intersection $\leftarrow$ 0
  union $\leftarrow$ 0
  i $\leftarrow$ 0
  j $\leftarrow$ 0
  while $b1[i] \neq$ -1 and $b2[j] \neq$ -1 **do**
    if $b1[i] = b2[j]$ **then**
      intersection $\leftarrow$ intersection + 1
      i $\leftarrow$ i + 1
      j $\leftarrow$ j + 1
    end if
    if $b1[i] < b2[j]$ **then**
      i $\leftarrow$ i + 1
    end if
    if $b1[i] > b2[j]$ **then**
      j $\leftarrow$ j + 1
    end if
    union $\leftarrow$ union + 1
  end while
  union $\leftarrow$ union + 1
  return $\frac{intersection}{union}$
end procedure

Algorithm 2 minHashSig Calculation

procedure MINHASHSIG($termsBinaryRepresenatation$)
  Create minHash array in size of numPermutations
  for File f $\leftarrow$ termsBinaryRepresenatation.KeySet() **do**
    bins $\leftarrow$ termsBinaryRepresenatation.get(f)
    for Permutation p $\leftarrow$ permutationses **do**
      min $\leftarrow$ Integer.MaxValue
      for Boolean b $\leftarrow$ bins **do**
        if b is $true$ **then**
          if p.(b) $<$ min **then**
            min $\leftarrow$ p.(b)
          end if
        end if
      end for
    end for
    Store min in minHash
  end for
  return minHash
end procedure

Algorithm 3 Approximate Jaccard Calculation

procedure APPROXIMAT JACCARD($file1, file2$)
    b1 $\leftarrow minHash$ of file1
    b2 $\leftarrow minHash$ of file2
    intersection $\leftarrow 0$
    for each pair $< d1, d2 >$ in b1 b2 **do**
        if d1 is $true$ and d2 is $true$ **then**
            intersection $\leftarrow$ intersection + 1
        end if
    end for
    return $\frac{intersection}{numPermutations}$
end procedure

Algorithm 4 MinHash Matrix Calculation

procedure MINHASH MATRIX($minHash$)
    matrix $\leftarrow$ [numPermutations][min.Hash.Keyset.size]
    Iterator $\leftarrow$ $minHash$'s iterator
    for each string in Iterator **do**
        hashs $\leftarrow$ iterator.next
Foreach permutationse in permutationses
        matrix $\leftarrow$ hashs[numPermutation]
    end for
    return matrix
end procedure

MinHashAccuracy

Table 1: Error is row, Number of Permutation is column. Data is (Large Error Pairs, Exact Jac in Sec, Appro Jac in Sec)

	0.04	**0.07**	**0.09**
400	(3131,0.527, 0.199)	(8, 0.535, 0.184)	(1, 0.565, 0.189)
600	(658, 0.52, 0.268)	(2, 0.535, 0.268)	(1, 0.563, 0.279)
800	(267, 0.532, 0.361)	(4, 0.542, 0.358)	(1, 0.573, 0.368)

In conclusion,Increasing number of permutation result in less error, however, it does not bring a large increasing of time complexity. It is because the increasing number is small to the number of terms. For example, It is no large time complexity different between calculation on 400 rows and 600 rows by computer.

Another point is that, the large error pairs are in the range of [0.04, 0.07], since when we adjust the allowed error to 0.09, there is only 1 pair in that range. The error happens because of the permutation we picked may not be able to totally permute the binary representation.