ComS 535x:
Project Report #4

Due on May 1, 2015

Instructor: Professor Aduri, Pavankumar

Yuanyuan Tang, Chenguang He

Contents

Algorithm for building index

Algorithm 1 Index Algorithm

```
procedure BUILD INDEX($Filef$)
    for $Wordwinf$ do
        $w \leftarrow modify(w)$
        if $wisinInverted$ then
            $i \leftarrow Inverted.get(w)$
            if the last document in i.postringpart is current document then
                $i.postringpart.TFtd \leftarrow i.postringpart.TFtd + 1$
            else
                $i.postringpart.document \leftarrow f$
                $i.dictionaryPart.DFt \leftarrow i.dictionaryPart.DFt + 1$
            end if
        else
            $i = newInvertedIndex$
            $i.postingpart \leftarrow (f, i)$
        end if
    end for
end procedure
```

Data structures used for the index (for both dictionary and postings part)

I build three classes, InvertedIndex, DictionaryPart and PostingsPart.

In InvertedIndex, it consist of a array with PostingsPart and one DictionaryPart. In PostingsPart it consist of a integer representation of the document, which the order of index is as same as in the files array, a TFtd. In DictionaryPart, it consist of a string of current term and the DFt.

The reason of using arraylist to store postingpart is that, we need to use the order of index to set the index increasing when we build the index. Also, the order is used to build the weight vector.

Algorithm for building the length of document vectors

Algorithm 2 length of document vector Algorithm

```
procedure BUILD LENGTH($WeightedVector weightedvector$)
    len[weightedvector] ← empty array
    for w ∈ weightedvector do
        sum ← 0
        for n ∈ w do
            sum += $n^2$
        end for
        len.add($sqrt(sum)$)
        for Index i of n ∈ w do
            set i th element to n/ sqrt(sum)
        end for
    end for
    return len
end procedure
```

Output of two queries

Query1: “astronomers radio instruments adaptive optics”.
Output is:

```
The top 10 document with its cosine similarity to q is :
the top 1 rank document is space-989.txt-clean , the cosine similarity is 0.2622889780894616
the top 2 rank document is space-987.txt-clean , the cosine similarity is 0.166600487661478
the top 3 rank document is space-985.txt-clean , the cosine similarity is 0.13831364844821636
the top 4 rank document is space-979.txt-clean , the cosine similarity is 0.13342135614608228
the top 5 rank document is space-759.txt-clean , the cosine similarity is 0.11305853420375138
the top 6 rank document is space-744.txt-clean , the cosine similarity is 0.11204706162701411
the top 7 rank document is space-739.txt-clean , the cosine similarity is 0.09987638192931692
the top 8 rank document is space-586.txt-clean , the cosine similarity is 0.09346295409828784
the top 9 rank document is space-492.txt-clean , the cosine similarity is 0.08698815821010417
the top 10 rank document is hockey719.txt , the cosine similarity is 0.08566037212705577
```

Query2:“Mattias Timmander MoDo elite league team”.
Output is:

```
The top 10 document with its cosine similarity to q is :
the top 1 rank document is hockey123.txt , the cosine similarity is 0.2933688991706478
the top 2 rank document is hockey125.txt , the cosine similarity is 0.1548387900355622
the top 3 rank document is hockey884.txt , the cosine similarity is 0.10925888803206889
the top 4 rank document is baseball811.txt , the cosine similarity is 0.05071168274469853
the top 5 rank document is hockey957.txt , the cosine similarity is 0.07410033028289059
the top 6 rank document is hockey995.txt , the cosine similarity is 0.06454051757187494
the top 7 rank document is baseball734.txt , the cosine similarity is 0.06364779113409008
the top 8 rank document is hockey106.txt , the cosine similarity is 0.06022956762946571
the top 9 rank document is hockey520.txt , the cosine similarity is 0.05878079784709665
the top 10 rank document is hockey159.txt , the cosine similarity is 0.05490325657686845
```