CSC 583 Homework 1

Sina Ehsani

January 25, 2019

1 Problem 1

Consider these documents:
Doc 1 breakthrough drug for schizophrenia
Doc 2 new approach for treatment of schizophrenia
Doc 3 new hopes for schizophrenia patients
Doc 4 new schizophrenia drug

1. Draw the term-document incidence matrix for this document collection.

	Doc 1	Doc 2	Doc 3	Doc 4
approach	0	1	0	0
breakthrough	1	0	0	0
drug	1	0	0	1
for	1	1	1	0
hopes	0	0	1	0
new	0	1	1	1
of	0	1	0	0
patients	0	0	1	0
schizophrenia	1	1	1	1
treatment	0	1	0	0

2. Draw the inverted index representation for this collection, as in Figure 1.3 in IIR.

$$\boxed{\text{approach}} \longrightarrow \boxed{2} \tag{1}$$

$$\boxed{\text{breakthrough}} \longrightarrow \boxed{1} \tag{2}$$

$$\boxed{\text{drug}} \longrightarrow \boxed{1}\boxed{4} \tag{3}$$

$$\boxed{\text{for}} \longrightarrow \boxed{1}\boxed{2}\boxed{3} \tag{4}$$

$$\boxed{\text{hopes}} \longrightarrow \boxed{3} \tag{5}$$

$$\boxed{\text{new}} \longrightarrow \boxed{2}\boxed{3}\boxed{4} \tag{6}$$

$$\boxed{\text{of}} \longrightarrow \boxed{2} \tag{7}$$

$$\boxed{\text{patients}} \longrightarrow \boxed{3} \tag{8}$$

$$\boxed{\text{schizophrenia}} \longrightarrow \boxed{1}\boxed{2}\boxed{3}\boxed{4} \tag{9}$$

$$\boxed{\text{treatment}} \longrightarrow \boxed{2} \tag{10}$$

3. What are the returned results for these queries:

(a) schizophrenia AND drug

$$1111\ AND\ 1001 = 1001$$

(b) for AND NOT(drug OR approach)

1.drug or approach:

$$1001\ OR\ 0100 = 1101$$

2.NOT(drug OR approach):

$$0010$$

3.for AND NOT(drug OR approach)

$$1110\ AND\ 0010 = 0010$$

2 Problem 2

1. Write out a postings merge algorithm, in the style of Figure 1.6 in IIR, for an x OR y query.

```
union(p1,p2)
while p1 != NIL and p2 != NIL
do if docID(p1) = docID(p2)
  then ADD(answer, docID(p1))
  p1 <- next(p1)
  p2 <- next(p2)
else if docID(p1) < docID(p2)
  then ADD(answer, docID(p1))
  p1 <- next(p1)
else
  then ADD(answer, docID(p2))
  p2 <- next(p2)

# When only one of the tokens is available in all of the documents:
while p1 != NIL
  ADD(answer, docID(p1))
  p1 <- next(p1)

while p2 != NIL
  ADD(answer, docID(p2))
  p2 <- next(p2)
return(answer)
```

2. Write out a postings merge algorithm, in the style of Figure 1.6 in IIR, for an x AND NOT y query.

```
# p1 AND NOT p2

andnot(p1,p2)
while p1 != NIL and p2 != NIL
do if docID(p1) = docID(p2)
  p1 <- next(p1)
  p2 <- next(p2)
else if docID(p1) < docID(p2)
  then ADD(answer, docID(p1))
    p1 <- next(p1)
else
  p2 <- next(p2)

# When p2 is not available in the documents:
while p1 != NIL and p2 = NIL
  ADD(answer, docID(p1))
  p1 <- next(p1)
return(answer)
```

3 Problem 3

Recommend a query processing order for:
(tangerine OR trees) AND (marmalade OR skies) AND (kaleidoscope OR eyes) given the following postings list sizes are shown in the assignment1.

As mentioned in the book, we will have to get the frequencies for all terms, and then estimate the size of each OR by the sum of the frequencies of its disjuncts. We can then process the query in increasing order of the size of each disjunctive term.

So we will start with the following and continue to the end. The number in front of each term is the sum of the frequencies of its disjuncts.

1. kaleidoscope OR eyes = 259, 965
2. marmalade OR skies =282, 449
3. tangerine OR trees = 403, 821

4 Problem 4

How should the Boolean query x OR NOT y be handled? Why is the naive evaluation of this query normally very expensive? Write out a postings merge algorithm that evaluates this query efficiently.

The naive evaluation is expensive, because you have to seek all the documents twice. First finding all the documents where y is not included and then do the OR query again on all the documents two find the x OR NOT y query. This implementation will have a runtime of 2 * N, where N is the number of docs in the collection. Using the following algorithm, we can reduce the time complexity:

```
# p1 OR NOT p2
ornot(p1,p2):
m=1

while p1 != NIL and p2 != NIL
  p2 <- 0 # We start with 0, this will force the algorithm to count all the documents before either p1 or p2

  if docID(p2) < docID(p1)
    for i in [docID(p1)-docID(p2)]
      while docID(p2+i) < docID(next(p2))
        ADD(answer, docID(p2+i))
        i++
    p2 <- next(p2)
  else
    ADD(answer, docID(p1))
    p1 <- next(p1)

# When only one of the tokens is available in all of the documents:
while p1 != NIL
  for i in count(docID)
    ADD(answer, docID(i))

while p2 != NIL
  while(docID(m)<docID(p2)
    ADD(answer, docID(m))
    m++
  p2 <- next(p2)

return(answer)
```