UCT CSC303 2005 :: XML/IR :: Exam [25 marks]

Answer Questions 1 AND 2.

Then answer 3 questions from among Questions 3-6.

(Questions 3 and 4 are optional – Sonia will provide the other questions).

Question 2 : Boolean Information Retrieval [10]

Consider the following collection of documents:

DocumentID	**Terms**
Doc1	one two three
Doc2	one one two two
Doc3	one
Doc4	three three three three

1. Build inverted files for this document collection. Include a term occurrence count with each DocumentID listed. [3]

One	*Doc1:1 Doc2:2 Doc3:1*
Two	*Doc1:1 Doc2:2*
Three	*Doc1:1 Doc4:4*

2. If the query "two three" is submitted to a Boolean-OR-based search engine, which results will remain after filtering? [1]

Doc1, Doc2, Doc4

3. Using the following ranking formula, compute a ranking value for each result from the previous question. [3]

$$Similarity = \frac{1}{|D|} \sum_{t \in Q \cap D} (1 + \log_e f_{d,t}).\log_e\left(1 + \frac{N}{f_t}\right)$$

Assume that:

- D is the length of the document, including all terms in the document.
- Only terms common to both query and document and considered.
- N = the total number of documents in the result set.
- $f_{d,t}$ = term frequency of term t in document d
- f_t = number of documents term t appears in.

Sim[Doc1,Q] = 1/√3 [(1+log 1)log(1+3/2) + (1+log 1)log(1+3/2)] = 1.058

Sim[Doc2,Q] = 1/√8 [(1+log 2)log(1+3/2)] = 0.5485

Sim[Doc4,Q] = 1/4 [(1+log 4)log(1+3/2)] = 0.5466

4. Discuss how the inverted files can be optimised to use less space if the number of documents (and therefore document identifiers) is large. [2]

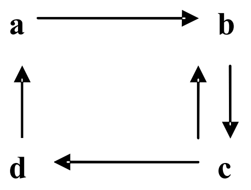
Use a differential encoding for the sorted document ids, with each value being the difference between the current id and the previous one [1] – then these values will be smaller numbers and can be stored using less actual space. [1]

5. Briefly discuss one technique that can be used to improve on recall. [1]

LSI [1] Stemming [1] ...

Question 3 : PageRank [10]

1. Apply the PageRank algorithm to the following link graph [6].



$$\begin{tikzcd} a \arrow[r] & b \arrow[d, shift left] \\ d \arrow[u] & c \arrow[l] \arrow[u, shift left] \end{tikzcd}$$

Start with equal ranks of 1/4 each. Stop after 4 iterations, or when C=5/16. Show all calculations, including N(umber of forward links) values and B(ack links) sets.

(Hint: Remember that $r[i]_n = \sum_{j \in B[i]} \frac{r[j]_{n-1}}{N[j]}$)

	N	*B*	*R[0]*	*R[1]*	*R[2]*	*R[3]*	*R[4]*
A	*1*	*D*	*¼*	*¼*	*1/8*	*1/8*	*3/16*
B	*1*	*A C*	*¼*	*3/8*	*3/8*	*5/16*	*5/16*
C	*2*	*B*	*¼*	*¼*	*3/8*	*3/8*	*5/16*
D	*1*	*C*	*¼*	*1/8*	*1/8*	*3/16*	*3/16*

One mark for N, one mark for B, one mark for each correct iteration

3. How does the HITS algorithm differ conceptually from the PageRank algorithm? [2]

HITS assigns importance to, and calculates ranking on the basis of, both pages with lots of links to them (authorities) and pages with lots of links to other pages (hubs) – PageRank only considers authorities. [2]

4. Contrast the runtime performance of simple PageRank with simple HITS. [2]

HITS calculates the focussed subgraph at runtime while PageRank calculates all ranks a priori – thus PageRank is much faster at runtime.

Question 4 : XML / XSLT [10]

1. Write an XSLT template to transform

```
<people>
<name age="21">wert</name>
<name age="23">polk</name>
```

```
    <name age=”22”>jik</name>
    </people>
```

into:

```
    <table>
    <tr><th>Names</th><th>Ages</th></tr>
    <tr><td>wert</td><td>21</td></tr>
    <tr><td>polk</td><td>23</td></tr>
    <tr><td>jik</td><td>22</td></tr>
    </table>
```

Assume that the values such as “wert” may differ from one document to another. Assume the source namespace prefix is “source” and the destination prefix is “dest”. Assume that the number of <name> tags is unbounded in its defining XML Schema. Use the following as a starting point. [5]

```
<xslt:template match=”source:people“>
. . .
</xslt:template>
```

```
<xslt:template match=”source:people”>
                                                  <dest:table>
  <dest:tr><dest:th>Names</dest:th><dest:th>Ages</dest:th></dest:tr>
  <xslt:for-each select=”source:name”>
    <dest:tr>
      <dest:td>
        <xslt:value-of select=”.”/>
      </dest:td>
      <dest:td>
        <xslt:value-of select=”@age”/>
      </dest:td>
    </dest:tr>
  </xslt:for-each>
  </dest:table>
</xslt:template>
```

Minus one mark for each major error or half for minor or repeated errors.

2. Write an XML Schema type definition corresponding to the contents of the people node. [5]

```
  <complexType>
    <sequence>
      <element name="name" minOccurs="0" maxOccurs="unbounded">
```

```
        <complexType>
          <simpleContent>
             <extension base=”string”>
               <attribute name="age" type="string" use=”required”/>
             </extension>
           </simpleContent>
        </complexType>
      </element>
    </sequence>
  </complexType>
```
Minus one mark for each major error or half for minor or repeated errors.