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

Question 2 : Information Retrieval [10]

Consider the following collection of documents:

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

1. Draw a term×document matrix of occurrence counts for this document collection. [3]

	Doc 1	*Doc 2*	*Doc 3*	*Doc 4*
One	*2*	*2*	*1*	*4*
Two	*1*	*2*	*1*	
Three	*1*			

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

Doc1, Doc2, Doc3

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/√6 [(1+log 1)log(1+3/3) + (1+log 1)log(1+3/1)] = 0.8489

Sim[Doc2,Q] = 1/√8 [(1+log 2)log(1+3/3)] = 0.4149

Sim[Doc3,Q] = 1/√2 [(1+log 1)log(1+3/3)] = 0.4901

4. The term×document matrix can be reduced in dimensionality using the LSI algorithm. Explain how the LSI algorithm achieves this reduction in space utilisation. [2]

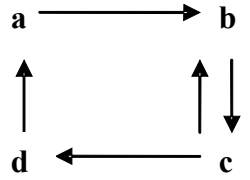
The td matrix is decomposed using SVD. [1] Then, the relatively insignificant components in the diagonal matrix are discarded, resulting in a reduction in the S and D matrices as well. [1]

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

LSI [1] Relevance feedback [1] …

Question 5 : HITS [10]

1. Apply the HITS algorithm to the following focussed sub-graph resulting from a query [6].



Start with equal ranks of 1/4 each for each of the hub and authority values. Stop after 2 iterations of each of a and h (not including the initial values), or when a2[A] =5/15. Show all calculations, including F(orward links) and B(ack links) sets.

Basic algorithm:

- authority value is the sum of all hub values of pages pointing to it
- the hub value is the sum of all authority values of pages it points to
- both values must be normalised after each iteration

	F	*B*	*a[0]*	*h[0]*	*a[1]'*	*a[1]*	*h[1]'*	*h[1]*	*a[2]'*	*a[2]*	*h[2]'*	*h[2]*
A	*B*	*D*	*¼*	*¼*	*¼*	*1/5*	*2/5*	*2/7*	*1/7*	*1/10*	*5/10*	*5/15*
B	*C*	*A C*	*¼*	*¼*	*½*	*2/5*	*1/5*	*1/7*	*5/7*	*5/10*	*1/10*	*1/15*
C	*B D*	*B*	*¼*	*¼*	*¼*	*1/5*	*3/5*	*3/7*	*1/7*	*1/10*	*8/10*	*8/15*
D	*A*	*C*	*¼*	*¼*	*¼*	*1/5*	*1/5*	*1/7*	*3/7*	*3/10*	*1/10*	*1/15*

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. The Open Directory Project maintains a directory of websites, each classified into a specific category. How would this project use HITS authority and/or hub values if it were to build a search engine of its categories? How would this be different if the search engine was meant to find websites in the categories instead of the categories themselves? [2]

In the first instance, we want to locate hubs so we use mainly the hub values in ranking. [1] The the second case, we want to find authoritative pages linked from the hubs so we use mainly the authority values in ranking. [1]

Question 6 : XML / XSLT [10]

1. Write an XSLT template to transform

```
<htmj>
                                                    <heading>
                              <tr><th>Names</th><th>Ages</th></tr>
                                                  </heading>
                                                    <table>
                            <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>
</htmj>
```

into:

```
<people>
<name age="21">wert</name>
<name age="23">polk</name>
<name age="22">jik</name>
</people>
```

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 <tr> tags is unbounded in its defining XML Schema. Use the following as a starting point. [5]

```
<xslt:template match="source:htmj">
. . .
</xslt:template>
```

<xslt:template match="source:htmj">

<dest:people>

<xslt:for-each select="source:table/source:tr">

<dest:name age="{source:td[2]}">

<xslt:value-of select="source:td[1]"/>

</dest:name>

</xslt:for-each>

</dest:people>

</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 table node. Assume that both names and ages are simple strings. [5]

```
<complexType>
  <sequence>
    <element name="tr" minOccurs="0" maxOccurs="unbounded">
      <complexType>
        <sequence>
            <element name="td" type="string" maxOccurs=”2” minOccurs=”2”/>
        </sequence>
      </complexType>
    </element>
  </sequence>
</complexType>
```

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