CSCE 561 Assignment #1, Fall 2012

Dr. Raghavan
Assigned: September 6, 2012
Due: September 24, 2012

Note:
1. You must show all details of work for each question
2. Staple the question and answer sheet together
3. Make a cover with Name, CLID
4. Number all pages and give an index to each question
*5. Most importantly, any sort of cheating will **NOT** be tolerated. More information can be found on class Web page on cheating policy.*

Q1. (20 points) The definition in the fuzzy-set theory for ∩ and ∪ in terms of min and max, verify whether the following set-theoretic axioms hold:

a). $(A \cap B) \cup C = (A \cup C) \cap (B \cup C)$

b). $\neg (A \cup B) = \neg A \cap \neg B$

c). $A \cap (A \cup B) = A$

Q2. (20 points)

	t_1	t_2	t_3	t_4	t_5
d_1	0.3	0.5	0.0	0.6	0.6
d_2	0.8	0.6	0.3	0.5	0.3
d_3	0.6	0.0	0.8	0.1	0.7
d_4	0.8	0.2	0.5	0.6	0.3

Table 1

Table 1 represents a matrix of documents and terms. Each document is represented by a row. Each element in the row represents the weights of inference of corresponding terms. Compute the RSVs of the following queries for each of the documents d1 through d4

a). Assume $\wedge$ and $\vee$ are defined, respectively, as min and max.

(1). $\neg(t1 \wedge t3 \wedge t5) \vee (\neg t2 \wedge \neg t4) \vee (t1 \wedge \neg t2)$
(2). $\neg(t1 \wedge t2 \wedge t4) \vee (t3 \wedge t5)$

b). Suppose $\vee$ and $\wedge$ are defined as 'bold' combination-

operator

$a \vee b = \min(a+b, 1)$,

$a \wedge b = \max(0, a+b-1)$,

where a and b membership function values of an element x belonging to fuzzy sets A and B. For this part, use **0.5 level fuzzy sets**.

(1). $t1 \wedge \neg t4$
(2). $t3 \vee (t2 \wedge t4) \vee (\neg t1 \wedge t5)$

Q3. (60 points)

Introduction

This question is based on the notion of document retrieval using Vector Space Model. The basic concept is that the relationships between the terms and the documents they contain can be represented as a term-document matrix of size MxN, where M is the number of terms and N is the number of documents. In the matrix, the (i,j)-th element is the 'term weight' of the i-th term on the j-th document. There already exist quite a few different algorithms to calculate the 'term weight', such as the normalized frequency, the tf-idf weight, and so on.

The normalized frequency is the weight of the term in the document and is calculated as

$$tf_{ij} = \frac{f_{ij}}{\max_k(f_{kj})},$$

where $1 \leq k \leq M$.

Each column in the term-document matrix corresponds to one document, which can be represented as an M-dimensional vector with the term weight as its components, such as

$\overrightarrow{d_1}$ = (0, 0.2, 0.5 … 1.0). A query q1 can also be seen as a vector $\overrightarrow{q1}$. Then, the RSV of q1 on d1 can be calculated as $RSV = \frac{\overrightarrow{d_1} \cdot \overrightarrow{q_1}}{|\overrightarrow{d_1}| * |\overrightarrow{q_1}|}$ (the inner product of $\overrightarrow{d_1}$ and $\overrightarrow{q_1}$),

where $|\overrightarrow{d_1}|$ and $|\overrightarrow{q_1}|$ are the Norm of vector $\overrightarrow{d_1}$ And $\overrightarrow{q_1}$, such as $|\overrightarrow{d_1}| = \sqrt{0^2 + 0.2^2 + 0.5^2 + ... + 1.0^2}$.

Copy the files from http://www.cacs.louisiana.edu/~cmps561/Assign1_Q3.zip. The directory has four text documents (d1.txt, d2.txt, d3.txt, and d4.txt), one stop-word file (stopwords.txt) and four java classes. This documents collection is the input of the program of part (a). The 'stopwords.txt' will be used to remove the stop-words in your implementation. The java classes are to be used to create the term-document matrix having normalized frequencies, $tf_{ij,}$ as its elements.

Implementation

Based on the above documents collection and algorithms, do the following implementations.

(a). Remove stop words and stem the documents collection with the Porter's Stemming algorithm. Create the basic term-document matrix of size MxN, where the (i,j)-th element is frequency of term ti within document dj (viz. fij).

the Java classes in the folder you download above with the following command:

```
java DocVector d1.txt "\!\?\[\].,;-" stopwords.txt > DocVector_d1.txt
```

where d1.txt is the input document to be processed, DocVector_d1.txt is the resulting table from d1.txt. In the result table, each line represents (word, frequency and normalized frequency). The first line of the output file is .I followed by the document name. Write a program to read data of all the four result documents to form the basic term-document matrix and the normalized frequency term-document matrix. Use the latter matrix for part (b).

(b). For any given query, your system should be able to compute the RSVs of all the documents and present the results to users. Users can input queries on your interface and the result documents should be shown in a decreasing order according to the RSV value. The queries will be vectors with a number between 0 and 1 as its components, such as

$\vec{q}_1 = (0.6,0.3,08....0.0)$

More details about normalized frequency, Porter's Stemming, and RSV calculation can be found in Chapter #2 of Lingras's book.

(c). An **inverted index** is a file containing a list of keywords. Generally the keywords are in sorted order. Each keyword is associated with a list of documents containing it together with the term frequencies. The keywords are called dictionary and their occurrences information is called postings. When you output the inverted index file, each line in the output should contain a keyword, and the format of the inverted index is:

<keyword> <docid>:<frequency> <docid>:<frequency> … <docid>:<frequency>

where

- keyword: the keyword
- docid: the unique ID of the document containing the keyword
- frequency: the frequency of the keyword in that document
- keyword and (document ID, frequency) pairs are separated by one space

Example:

book 4:3 6:7 10:2

pen 1:1 2:6 7:4 8:2 9:8

Choosing an appropriate data structure for inverted index is important for issues like storage requirement, fast accessing the index, updating the index etc. See chapter 1 of the book "Introduction to Information Retrieval" (first reference in the class web page). For example, a fixed-length array would be wasteful as only a few keywords

occur in many documents, while very few documents contain many keywords. Variable length data structures like Java Hash map, Array List or Linked list can be good options. The various keywords in the dictionary should be maintained in a sorted order.

Write a program to create and output the inverted index from the basic term-document matrix computed in part (a).

You can implement your design using any programming language (i.e., C++, PERL, Java, VB, C#), any technology (JSP, Java Servlets, ASP, ASP.net etc)

Evaluation

Stemming and removing stop words, creating the basic document-term matrix and the document-term matrix with normalized frequency (20 points)
Calculating RSV (15 points)
Creating and displaying the Inverted Index (20 points)
Error messages (5 points)
Total: 60 points