CIS 833 – Information Retrieval and Text Mining **Name:_______________**

Homework Assignment 2 (20 points) – due September 25th (by midnight)

Note: Please remember that you are allowed to discuss the assigned exercises, but you should write your own solution.

Exercise 1 (Boolean Model) **[3 points]** Which of the documents in the table below will be retrieved given the Boolean query: (chaucer AND (NOT swift)) OR ((NOT chaucer) AND (swift OR shakespeare))

	Chaucer	Milton	Shakespeare	Swift
D1	0	0	0	0
D2	0	0	0	1
D3	0	0	1	0
D4	0	0	1	1
D5	0	1	0	0
D6	0	1	0	1
D7	0	1	1	0
D8	0	1	1	1
D9	1	0	0	0
D10	1	0	0	1
D11	1	0	1	0
D12	1	0	1	1
D13	1	1	0	0
D14	1	1	0	1
D15	1	1	1	0
D16	1	1	1	1

Solution:

Documents that satisfy (chaucer AND (NOT swift)) are D9, D11, D13, and D15. Documents that satisfy ((NOT chaucer) AND (swift OR shakespeare)) are D2, D3, D4, D6, D7, and D8. As these two expressions are linked by OR in the query, the documents that will be retrieved as a result of this query are obtained by taking the union of the above two sets. So, the documents retrieved are:

D2, D3, D4, D6, D7, D8, D9, D11, D13, D15

Exercise 2 (Vector Space Model) **[7 points]** Given a query Q and a collection of documents A,B,C, represented as a document-word count matrix:

	Cat	Food	Fancy
Q	3	4	1
A	2	1	0
B	1	3	1
C	0	2	2

1. Compute the tf .idf representation of the query and of the documents (use log base 2).
2. Compute the cosine similarity of each document to the query Q.

Solution: We need to calculate idf's for all terms, tf's for all terms/all documents and length for all documents.

idf table (based on documents A,B,C)

Cat	Food	Fancy
log (3/2) = 0.585	log (3/3) = 0	log (3/2) = 0.585

tf table (term frequency in a document, normalized by the frequency of the most common term in that document)

	Cat	Food	Fancy
Q	3/4 = 0.75	4/4 = 1	1/4 =0.25
A	2/2 = 1	1/2 = 0.5	0/2 = 0
B	1/3 = 0.33	3/3 = 1	1/3 = 0.33
C	0/2 = 0	2/2 = 1	2/2 = 1

tf*idf table

wij	Cat	Food	Fancy
Q	0.75 * 0.585 = 0.438	1 * 0 = 0	0.25 * 0.585 = 0.146
A	1 * 0.585 = 0.585	0.5 * 0 = 0	0 * 0.585 = 0
B	0.33 * 0.585 = 0.193	1 * 0 = 0	0.33 * 0.585 = .193
C	0 * 0.585 = 0	1 * 0 = 0	1 * 0.585 = 0.585

Document length table:

	Q	A	B	C
Length	0.461	0.585	0.272	0.585

Cosine similarity between documents and query:

	A	B	C
cosSim(Q,Doc)	0.950	0.898	0.316

Exercise 3 (Retrieval using an Inverted Index) [10 points]

i. Construct an inverted index for the following collection of documents. Show the index graphically with linked lists. Your index terms should be stemmed and should not be in the stop words list.

```
Doc 1. John gives a book to Mary
Doc 2. John who reads a book loves Mary
Doc 3. Who does John think Mary love?
Doc 4. John thinks a book is a good gift
```

```
Term

john  -> 4 – D1,1 – D2,1 – D3,1 – D4,1
mary  -> 3 – D1,1 – D2,1 – D3,1
give  -> 1 – D1,1
book  -> 3 – D1,1 – D2,1 – D4,1
read  -> 1 – D2,1
love  -> 2 – D2,1 – D3,1
think -> 2 – D3,1 – D4,1
gift  -> 1 – D4,1
good  -> 1 – D4,1
```

Note: we construct the index incrementally as we parse the documents in the given collection. The index is stored in a hashtable that has the vocabulary terms as keys and the DF followed by posting lists as values.

ii. What other information can be pre-computed offline, in addition to the actual index?

The following information can be also precomputed offline:

- First, the number of documents can be computed into a variable N, as we parse the collection of documents. Initially, N = 0. For any new document in the collection, I increment N by 1.
- Second, we can compute max frequency for each document in another hashtable for which documents are keys and the max frequencies are the values. Every time we update the count for a term in a document, we compare the current count with the max frequency for that document and update the max frequency, if it happens to be smaller than the current count.
- Third, we can also compute length for each document offline.

iii. Show how you can compute document lengths incrementally by parsing the index. Remember to make use of a hashtable.

We can compute length of documents incrementally, by using yet another hashtable for which documents are keys and their length is the value. The length hashtable is computed with a second pass through the data, or equivalently with a pass of the inverted index. As I encounter a document j in the posting list of a term i in the inverted index, I increment the current length (originally 0) for that document with the product w_ij^2 (where w_ij is a TF-IDF weight for the term i in document j).

iv. Consider the query "`love Mary`" and simulate the retrieval of documents in response to this query. Show how the inverted index is used to identify relevant documents and how the cosine similarity between the query and the relevant documents is calculated incrementally using a hashtable.

Note: you don't need to calculate the exact number for this problem; instead, you should explain how each step is done, using the sample documents.

We need to parse the words in the query. We first find the word "love" – we search it in the index and we find that the word appears in 2 documents, D2 and D3. We create entries for D2 and D3 in a hashtable and use the information in the index to update the hashtable entries for these documents.

We have

D2 -> $w_{2,love}*w_{q,love}$
D3 -> $w_{3,love}*w_{q,love}$
Where $w_{doc,word}$ =[TF(word,doc)/max-freq(doc)]* log [N/IDF(word)]

The next word in the document is "mary" – this word appears in documents D1, D2 and D3. We update the hashtable entries for D2 and D3 and we create an entry for D3. We have:

D2 -> $w_{2,love}*w_{q,love} + w_{2,mary}*w_{q,mary}$
D3 -> $w_{3,love}*w_{q,love} + w_{3,mary}*w_{q,mary}$
D1 -> $w_{1,mary}*w_{q,mary}$

The final values need to be normalized by the product of document and query lengths.