CIS 833, Fall 2015 **Name:____________________**

Exam 1 – 75 minutes

This test consists of five questions. The number of points for each question is shown below.

- Read all questions carefully before starting to answer them.
- Write all your answers on the space provided in the exam paper.
- The order of the questions is arbitrary, so the difficulty may vary from question to question. Don't get stuck by insisting on doing them in order.
- Show your work. Correct answers without justification will not receive full credit. However, also be concise. Excessively verbose answers may be penalized.
- Clearly state any simplifying assumptions you may make when answering a questions.
- **Be sure to write your name on the test paper.**

Question	1	2	3	4	5	total
Points	20	20	20	20	20	100
Your points						

I. MapReduce [20 points]

To improve its recommendations, Amazon collects data about book reviews in files that contain records of the form:

```
(user, book, review, rating)
```

You will help Amazon with the recommendation process by writing a few MapReduce jobs for them.

i. Your first task is to write a MapReduce job to identify the average rating for each reviewed book. The input to the mapper will be files containing records in the format above. The output of the reducer will be key-value pairs of the form `book, avg(rating)`.

```
class MAPPER
    method INITIALIZE
        S<-new AssociativeArray
        C<-new Associative Array

    method MAP(record_id, record_content)
        book <- get_book_from_record
        rating <- get_book_rating
        S(book)<-S(book)+rating
        C(book)<-C(book)+1

    method CLOSE
        for all books
            EMIT (book, pair(S(book),C(book)))

class REDUCER
    method REDUCE (book, pairs [(s1,c1), (s2,c2),...])
        sum<-0
        cnt<-0
        for each pair (s,c) from pairs [(s1,c1), (s2,c2),...] do
            sum<-sum+s
            cnt<-cnt+c
        avg<-sum/cnt
    EMIT (book,avg)
```

ii. Your second task is to use the tabulation of `book,avg(rating)` in i. to find the books with the highest overall rating (highest average). In your solution, you should use the fact that MapReduce sorts by intermediate keys into the reducer groups.

```
//we are reading the file produced in i. – i.e., each line contains book, avg
method MAP(line_id, line_content)
split_line_content_into_book_and_avg
EMIT (avg, book)

//MapReduce will sort by intermediate keys –> avg, book pairs will be sorted by avg if only
//one reducer is used
method REDUCE (avg, list [book1,book2,...])      //intermediate key,value pairs
    // reduce is the identity function
    for each book in list [book1,book2,...]
        EMIT (book, avg)
```

iii. Your last task is to compute the number of times two books were both reviewed by the same user. The input will be the original files with records of the form `(user, book, review, rating)` and the output will be triples of the form `(book1, book2, count)`. For example, if we know that `user1` has reviewed both `book1` and book2, and `user2` has also reviewed both `book1` and `book2`, and there are no other users who have reviewed both `book1` and `book2`, then the answer will be (`book1`, `book2`, 2). You can assume that a user does not review the same book twice. Hint: you will need to write two MapReduce jobs to solve this problem.

Job 1

```
method MAP(record_id, record_content)
    user <- get_user_from_record
    book <- get_book_from_tuple
    EMIT (user, book)

method REDUCE (user, list of books reviewed by user)
    for all pairs of books reviewed by user
        order the pair alphabetically
        EMIT (book1, book2)
```

Job 2

```
method MAP(book1, book2)
    EMIT(<book1,book2>, 1)

map REDUCE(<book1,book2>, list [1,1,...])
    count = 0
    for all t in list
        count<-count+t
     EMIT (<book1,book2>, count)
```

II. Boolean Retrieval [20 points]

Assume the following collection of short documents.

```
Doc 1: banking on banks to raise the interest rate
Doc 2: jogging along the river bank to look at the sailboats
Doc 3: jogging to the bank to look at the interest rate
Doc 4: buzzer-beating shot banked in
Doc 5: scenic outlooks on the banks of the Potomac River
```

i. Construct a term-document matrix that can be used to perform Boolean retrieval. The index terms have already been arranged for you alphabetically in the following table:

Term	**Doc1**	**Doc2**	**Doc3**	**Doc4**	**Doc5**
bank	1	1	1	1	1
buzzer-beating	0	0	0	1	0
interest	1	0	1	0	0
jog	0	1	1	0	0
look	0	1	1	0	0
outlook	0	0	0	0	1
potomac	0	0	0	0	1
raise	1	0	0	0	0
rate	1	0	1	0	0
river	0	1	0	0	1
sailboat	0	1	0	0	0
scenic	0	0	0	0	1
shot	0	0	0	1	0

ii.What documents would be returned in response to the following queries?

bank ∧ ¬interest   Doc 2,4,5

interest ∧ jog ∧ ¬rate None

bank ∧(scenic ∨ jog) Doc 2,3,5

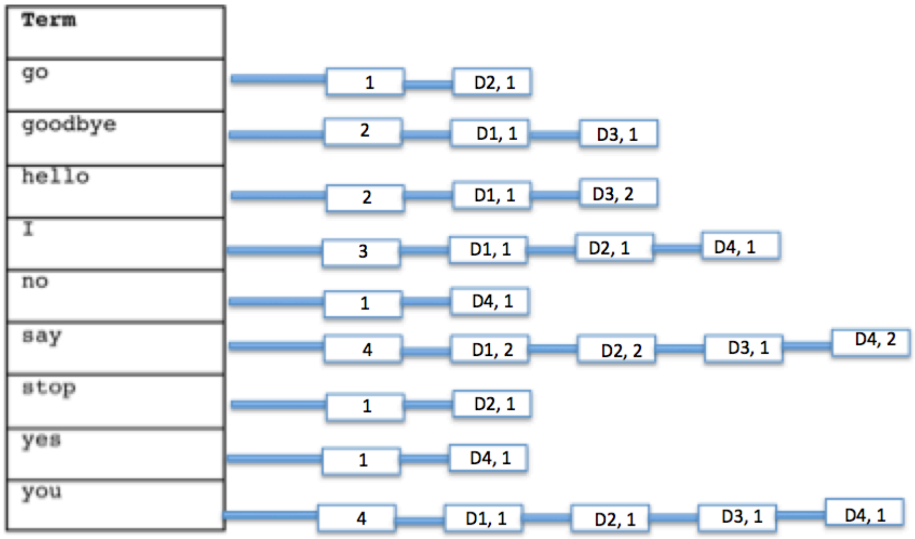
iii. Discuss two disadvantages of the Boolean retrieval model.

III. Inverted Index, TF-IDF and Cosine Similarity [20 points]

Consider the following collection of documents:

```
d1: You say goodbye, I say hello
d2: You say stop, I say go
d3: Hello, hello, you say goodbye
d4: I say yes, you say no
```

i. Construct an inverted index for this collection, assuming the vocabulary terms shown below. Show the index graphically with linked lists.



ii. Calculate the cosine similarity between d1 and d3, assuming the TF-IDF weighting scheme. Remember that the cosine between vectors d1 and d3 is given by the following formula:

$$\cos(\vec{d}_1,\vec{d}_3) = \frac{\vec{d}_1 \cdot \vec{d}_3}{|\vec{d}_1| \cdot |\vec{d}_3|} = \frac{\sum_{i=1}^{n}(w_{1i} \cdot w_{3i})}{\sqrt{\sum_{i=1}^{n} w_{1i}^2 \cdot \sum_{i=1}^{n} w_{3i}^2}}$$

	go	goodbye	hello	I	no	say	stop	yes	you
DF	1	2	2	3	1	4	1	1	4
IDF	log(4/1)	log(4/2)	log(4/2)	log(4/3)	log(4/1)	log(4/4)	log(4/1)	log(4/1)	log(4/4)
TF_1	0/2	1/2	1/2	1/2	0/2	2/2	0/2	0/2	1/2
TF_3	0/2	1/2	2/2	0/2	0/2	1/2	0/2	0/2	1/2

$$\cos(\vec{d}1,\vec{d}3)=\frac{w_{1,goodbye}*w_{3,goodbye}+w_{1,hello}*w_{3,hello}+w_{1,say}*w_{3,say}+w_{1,you}*w_{3,you}}{\sqrt{(w_{1,goodbye}^2+w_{1,hello}^2+w_{1,I}^2+w_{1,say}^2+w_{1,you}^2)*(w_{3,goodbye}^2+w_{3,hello}^2+w_{3,say}^2+w_{3,you}^2)}}$$

where $w_{i,term}$ is the TF*IDF weight of *term* in document I (obtained by multiplying IDF with TF in the table above)

iii. Consider the query "`say goodbye`" 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. (Don't worry about getting the exact numbers for this part, I just want to see how you use the inverted index data to get the cosine similarity).

We need to parse the words in the query. We first find the word "say" – we search it in the index and we find that the word appears in all 4 documents. We create entries for D1, D2, D3 and D4 in a hashtable and use the information in the index to update the hashtable entries for these documents.

We have

D1 -> $w_{1,say}*w_{q,say}$
D2 -> $w_{2,say}*w_{q,say}$
D3 -> $w_{3,say}*w_{q,say}$
D4 -> $w_{4,say}*w_{q,say}$
Where $w_{doc,word}$ =[TF(word,doc)/max-freq(doc)]* log [N/IDF(word)]

The next word in the document is "goodbye" – this word appears in documents D1 and D3. We update the hashtable entries for D1 and D3. We have:

D1 -> $w_{1,say}*w_{q,say} + w_{2,goodbye}*w_{q, goodbye}$
D3 -> $w_{3,say}*w_{q,say} + w_{3, goodbye} *w_{q, goodbye}$

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

iv. Explain how stemming typically affects recall.
Stemming typically increases recall because morphological variations of words are collapsed onto a single token, enabling retrieval of relevant documents that contain slight variations of the query tokens in addition to those that contain the query tokens themselves.

v. Explain the role of the IDF factor in standard term weighting.

The role of IDF is to decrease the weight of common terms that occur in many documents and therefore do not make useful, discriminative search terms and to upweight uncommon terms (ones that appear in few documents) since they are typically more specific and therefore better index terms.

IV. Latent Semantic Indexing [20 points]

Doc 1: dog cat cat
Doc 2: dog dog cat cat
Doc 3: dog dog dog dog cat cat
Doc 4: dog cat
Doc 5: dog dog cat
Doc 6: dog dog dog dog cat

i. Construct a term-document frequency matrix, X, for the given collection. The index terms have already been arranged for you alphabetically in the following table:

Term	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6
dog	1	2	4	1	2	4
cat	2	2	2	1	1	1

ii. The singular value decomposition of the term-document frequency matrix is shown below:

$$X = T_0 S_0 D'_0 = \begin{bmatrix} -0.88 & -0.48 \\ -0.48 & 0.88 \end{bmatrix} \times \begin{bmatrix} 7.31 & 0 \\ 0 & 1.88 \end{bmatrix} \times \begin{bmatrix} -0.25 & -0.37 & -0.61 & -0.19 & -0.30 & -0.54 \\ 0.68 & 0.42 & -0.09 & 0.21 & -0.04 & -0.55 \end{bmatrix}$$

We want to reduce the space from two dimensions ("dog" and "cat") to one dimension representing the concept "pets". How do we accomplish this? Show what matrices you need to multiply to obtain the new approximate document-term matrix $\hat{X}$.

We reduce the rank of matrix S_0 from 2 to 1 (i.e., keep only the diagonal element with higher value), and appropriately reduce matrices T_0 and D_0. The new approximate document term matrix will be TSD' which is:

$$\hat{X} = \begin{bmatrix} -0.88 \\ -0.48 \end{bmatrix} \times \begin{bmatrix} 7.31 \end{bmatrix} \times \begin{bmatrix} -0.25 & -0.37 & -0.61 & -0.19 & -0.30 & -0.54 \end{bmatrix}$$

iii. What is the vector representation for the query `q = {dog, dog, cat, dog}` in the original space (x_q) and in the reduced concept space (d_q)? (Remember that $d_q = x'_q T S^{-1}$).

In the original term space, we have $x_q=[1,1]'$

In the concept space, we have $d_q = x'_q T S^{-1} = \begin{bmatrix} 1 & 1 \end{bmatrix} \times \begin{bmatrix} -0.88 \\ -0.48 \end{bmatrix} \times \begin{bmatrix} \frac{1}{7.31} \end{bmatrix}$

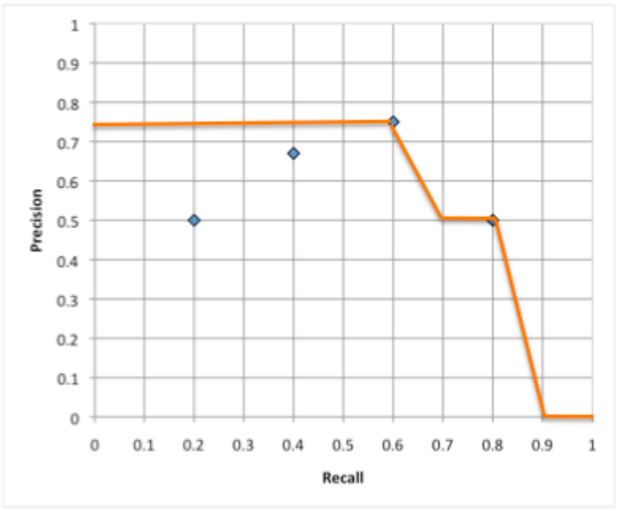
$= (-0.88-0.48) \times (1/7.31) = -0.18$

iv. Calculate the dot product similarity between the above query and Doc 1.

Dot product is $\langle d_q S, d_1 S \rangle = -0.18 \times 7.31 \times -0.25 \times 7.31$

V. IR Evaluation [20 points]

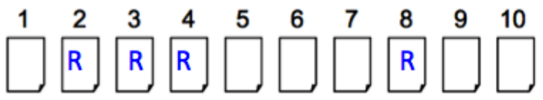
The following is a plot of uninterpolated precision-recall values for an IR system that retrieves 10 ranked documents when queried on a particular topic. You know that there are 5 relevant documents for this topic (but only 4 of them are retrieved by the IR system).



i. Draw the interpolated precision-recall curve in the plot above.

See above

ii. In the diagram below, each box represents a hit. Based on the above precision-recall plot, which hits are relevant? Write an "R" on the relevant hit. Leave the non-relevant hits empty.



We can check that when these documents are the 4 retrieved relevant documents, we get the numbers in the graph for the shown precision/recall points.

Document number	Recall	Precision
2	1/5 = 0.2	1/2 = 0.50
3	2/5 = 0.4	2/3 = 0.67
4	3/5 = 0.6	3/4 = 0.75
8	4/5 = 0.8	4/8 = 0.50

iii. Calculate the (uninterpolated) average precision for the query based on the graph above. You received full points also for correctly calculating interpolated average precision.

AP = (0.2 + 0.4 + 0.6 + 0.8) / 4