Text Information Retrieval, Mining, and Exploitation
CS 276A
Open Book Midterm Examination
Tuesday, October 29, 2002
Solutions

This midterm examination consists of 10 pages, 8 questions, and 30 points. It will form 20% of your final grade. We would like you to write your answers on the exam paper, in the spaces provided. To give you plenty of room, some pages are largely blank. If there isn't sufficient room, write on the back of a page, but please put an arrow or PTO on the front to tell us to look there. You have 70 minutes to complete the exam. Examinations turned in after the end of the examination period will either be penalized or not graded at all.

Stanford University Honor Code:

I attest that I have not given or received aid in this examination, and that I have done my share and taken an active part in seeing to it that others as well as myself uphold the spirit and letter of the Honor Code.

Name (printed): ____

Signature: ______________________ SUID: | | | | | | | | |

Question	Score	Possible
1 Eval		3
2 Index		4
3 Short		6
4 Query		6
5 T/F		3
6 Cosine		2
7 Phrases		3
8 LSI		3
Total		30

1. ***Evaluation: Recall and Precision (3pts)***

a. An IR system returns 3 relevant documents, and 2 irrelevant documents. There are a total of 8 relevant documents in the collection. What is the precision of the system on this search, and what is its recall?

The precision is given by tp/(tp+fp) = 3/5
The recall is given by tp/(tp+fn) = 3/8

b. Instead of using recall/precision for evaluating IR systems, we could use accuracy of classification. Consider a classifier (non-ranking IR system) that classifies documents as being either relevant or non-relevant. The accuracy of a classifier that makes *c* correct decisions and *i* incorrect decisions is defined as: *c*/(*c*+*i*).

(i) Why do the recall and precision measures reflect the utility (i.e., quality or usefulness) of an IR system better than accuracy does?

Ans.
An IR system which always returns no results will have high accuracy for most queries, since the corpus usually contains only a few relevant documents. Documents that are truly relevant are the only ones that will be mistakenly classified as nonrelevant, and thus the accuracy is close to 1. Recall and precision are two different measures that can jointly capture the tradeoff between returning more relevant results and returning fewer irrelevant results.

(ii) Suppose that we have a collection of 10 documents, and two different boolean retrieval systems A and B. Give an example of two result sets, A_q and B_q, assumed to have been returned by the system in response to a query *q*, constructed such that A_q has clearly higher utility and a better score for precision than B_q, but such that A_q and B_q have the same scores on accuracy.

Ans.
There are of course many correct answers. One simple correct answer is

Assume document 1 is the only relevant document.
$A_q = \{1,2,3\}$
$B_q = \{3\}$

Both A_q and B_q made 2 mistakes, so they have the same accuracy: 80%

The precision of A_q is 1/3, the precision for B_q is 0. Since B_q didn't return any relevant documents, it is of no utility.

2. ***Index size computation (4pts)***
Consider an index for 1 million documents each having a length of 1,000 words. Say there are 100K distinct terms in total. We do not wish to keep track of term-frequency information. What is the space requirement for:

a. An uncompressed term-document incidence matrix that does not exploit sparsity?

Ans.
n = 1M
m = 100K
The full bit-matrix including all zeros would need n X m entries:
10^6 X 10^5 bits = 10^{11} bits ~ 12 GB

b. Assuming a Zipf's Law distribution of terms, as in class, what is the space for the postings in an inverted index that encodes each docid with fixed-length binary codes (without the use of gaps or gamma encoding)?

Ans.
# of bits per code is given by $\left|\log_2 10^6\right| = 20$ bits
Space =

$$\sum_{k=1}^{100,000} 20 bits/code \times \frac{n}{k}$$
$$= 20 bits \times n \sum_{k=1}^{100,000} \frac{1}{k}$$
$$= 20 n \ln m \text{ bits}$$
$$= 20 \times 10^6 \ln 10^5 \text{ bits}$$

which is roughly 27.4 MB

c. Under the same assumptions, but with gamma encoding of the gaps between successive docIDs in the postings lists?

Ans.
This is very similar to problem #3 on the practice homework set. Please refer to those solutions for details. The difference is we sum from k = 1 to 100K, which leads to 17 groups (sum i = 1 to 17), with n set to 10^6. This leads to an upper bound answer of roughly 34.45 MB. A better bound is actually 23.8MB, which you approximate the $\sum_{k=2^{i-1}}^{2^i} \frac{1}{k}$ term as ln 2 instead of as 1.

3. ***Some short questions (6pts)***

a. If you wanted to search for s*ng in a Permuterm wildcard index, what key(s) would one do the lookup on?

Ans.
ng$s*

b. A crawler gathers documents and sends them to an indexer, which employs the following modules:
(A) a stemmer;
(B) a language detector to detect the language of each document;
(C) a stop-word eliminator
(D) a filter that detects the format (pdf, Word, etc.) of the document.

Give the correct sequence in which the indexer should apply these modules to a document:

Ans.
D B C A

c. We have a two-word query. For one term the postings list consists of the following 16 entries:

(4,6,10,12,14,16,18,20,22,32,47,81,120,122,157,180)

and for the other it is the one entry postings list [47]. Work out how many comparisons would be done to intersect the two postings lists, assuming:

(i) using standard postings lists

Ans.
11
(Compare 47 with entries 4 through 47)

(ii) using postings lists stored with skip pointers, with a skip length of $\sqrt{length}$, as recommended in class. Briefly justify your answer.

Ans.
6
(Compare 47 with entries 4,14,22,120,32,47)

(iii) Explain briefly how skip pointers could be made to work if we wanted to make use of gamma-encoding on the gaps between successive docIDs.

Ans.
Use absolute encodings rather than gap encodings for the target of skips.

d. Indexing New York Times newswire from 1991–1995 reveals that it contains about 400 million word tokens, and a lexicon of size about 1 million (given certain fixed decisions on term normalization, lowercasing, treatment of numbers etc.). What would be a good prediction of how many word tokens and what lexicon size one would get in indexing New York Times newswire from 1991–2000?

Ans.
Assuming the rate of creation remains the same, we can assume the number of tokens doubles. Thus, there are 800M tokens. Using the eqn. $V = k\sqrt{N}$ to model the original corpus of 400M tokens, with N=400M and V=1M, we solve for k, yielding k = 50. Using this value for k, and N=800M, we estimate that there are roughly $50 \times \sqrt{800M} = 1.4M$ tokens.

4. ***Retrieval models (6pts)***
Suppose we have a collection that consists of the 4 documents given in the table below. We will consider retrieval using two models:

a. Retrieval by a boolean semantics search engine.

b. Ranked retrieval by a probabilistic language model:
As discussed in Lecture 7, we use a mixture model between the documents and the collection, with both weighted at 0.5. Maximum likelihood estimation (mle) is used to estimate both as unigram models. (Document priors may be safely ignored.)

DocID	**Document text**
1	click go the shears boys click click click
2	click click
3	metal here
4	metal shears click here

a. With the Boolean model, what results will be returned for the query “(metal OR click) AND NOT here”?

Ans.
{1,2}

b. For the probabilistic language model (using the mixture model described above), work out the probabilities of the queries “click”, “shears”, and hence “click shears” according to each document, and use those probabilities to rank the documents returned by each query. Fill in these probabilities in the below table:

Ans.

Query	**Doc 1**	**Doc 2**	**Doc 3**	**Doc 4**
click	0.4688	0.7188	0.2188	0.3438
shears	0.1250	0.0625	0.0625	0.1875
click shears	0.0586	0.0449	0.0137	0.0645

The collection model for click is 7/16
The collection model for shears is 2/16=1/8

The entries above are a mixture of the collection model and the document model. E.g., the entry for {click,doc1} is simply 0.5 * 1/2 + 0.5 * 7/16 = 0.4688

Final ranking for “click shears” is:
Ans. 4,1,2,3

c. Using the calculations in **b.** as examples where appropriate, write one sentence each describing the treatment that such a probabilistic language model approach to IR give to:
(i) Term frequency in a document

Ans. Term frequency is included linearly in the model, but it is scaled linearly by the length of each document

(ii) Collection frequency of a term

Ans. Collection frequency influences the model because interpolation of the corpus model has the effect of narrowing the probability difference more for common terms – without smoothing documents 1 and 4 would score the same for the query *click shears*, but the smoothing gives the vote to document 4.

(iii) Document frequency of a term

Ans. Document frequency is not in the model, except in so far as collection frequency statistics serve as a partial surrogate

(iv) Length normalization of documents

Ans. Term frequencies are linearly scaled by the length of document: this is part of how document 4 wins on *click shears* with only 2 matching words, versus 5 for document 1.

5. ***Retrieval T/F (3pts)***

a. Mark these statements **true/false**:

Statement	True or False
(i) In a Boolean retrieval system, stemming never lowers precision.	**Ans.** F
(ii) In a Boolean retrieval system, stemming never lowers recall.	**Ans.** T/F (F if NOT operator is allowed)
(iii) Stemming increases the size of the lexicon.	**Ans.** F
(iv) Stemming should be invoked at indexing time but not while doing a query	**Ans.** F

b. Most IR systems are optimized for short queries. Consider a long query of more than 500 words, evaluated by a vector space model ranking IR system. Consider the following commonly employed optimizations and techniques. Does the technique help process **long** queries efficiently? Answer **true** if it does help, and **false** if it does not help, and explain your answer briefly in the space provided.

This technique helps long queries?	True or False
(i) Postings for each term ordered according to weight, largest weight first	**Ans.** T
Explain: **Ans.** This allows stopping processing of lists early, just as with short queries.	
(ii) Using an n-gram index	**Ans.** F
Explain: **Ans.** Long queries lead to many n-grams to lookup and merge (also, n-grams are generally not used with vector space models)	

6. ***Cosines and distances (2pts)***

Show that for length normalized vectors, ranking according to increasing order by Euclidean distance and ranking according to decreasing order by the cosine of the angle between the documents gives the same ranking. We are looking for a concrete answer, rather than intuition.

Ans.

There are several ways to solve this, depending on how much trigonometry you employ.

The goal is to show that $||u-v||$ increases as $1-\cos\theta$ increases. Equivalently, we can show $||u-v||^2$ is proportional to $1-\cos\theta$. The simplest proof uses the Law of Cosines:

$$||u-v||^2 = ||u|| + ||v|| - 2||u||||v||\cos\theta = 1 + 1 - 2\cos\theta = 2 - 2\cos\theta = 2(1 - \cos\theta)$$

(It is straightforward to derive the above, even if you didn't remember this Law, by expanding the summation represented by $||u-v||^2$)

7. *Phrase queries (3pts)*
Consider the following positional index in the form
<word:
docid1: position, offset, offset ... ;
offset_to_docid2: position, offset, offset ... ;
offset_to_docid3: position, offset, offset ... ;
etc.
>

<information:
11: 7, 18, 33, 12, 46, 31;
2: 3, 149;
54: 17, 11, 291, 30, 44;
5: 433, 67;
77: 54, 12, 3, 22;
19: 4, 12, 333;>
	<retrieval:
	11: 6, 20, 33, 72, 86, 231;
	3: 34, 19;
	53: 107, 191, 22, 40, 434;
	5: 363, 138;>

There are no skip pointers. We wish to process the phrase query *"information retrieval"* on this index.

a. Consider the elementary operation of adding an offset to an absolute position or docid to compute another absolute position or docid. What is the minimum number of these operations we must perform to find the **set of documents** satisfying this query? **(Subtractions are not allowed)**

Ans.

Docids: retrieval: 11 $\rightarrow$ 14 $\rightarrow$ 67 $\rightarrow$ 72 info: 11 $\rightarrow$ 13 $\rightarrow$ 67 $\rightarrow$ 72 **6 operations**	Positions: (for *information*, we start at 1, so we can do equality comparison with the corresponding position of *retrieval* to detect a match) 11: information: 1 $\rightarrow$ 8 $\rightarrow$ 26 retrieval: 6 $\rightarrow$ 26 67: information: 1 $\rightarrow$ 18 $\rightarrow$ 29 $\rightarrow$ 320 retrieval: 107 $\rightarrow$ 298 $\rightarrow$ 320 72: information: 1 $\rightarrow$ 434 $\rightarrow$ 501 retrieval: 363 $\rightarrow$ 501 **11 operations**

17 operations total

b. What are all the documents and all the absolute positions at which the query phrase occurs?

Ans. 11: {(25, 26), (58, 59)**}**, **67: {**(319, 320)**}**, **72: {**(500,501)**}**

8. ***LSI (3pts)***
Assume you have a set of documents consisting of 3 types, A, B and C:

A. English only documents
B. Spanish only documents
C. Documents for which there is both a Spanish and English version.

You want to be able to support cross-language retrieval; in other words, users with some information need should be able to issue queries in either English or Spanish, and retrieve documents of either language satisfying the information need.

The corpus is given below:

Type A:

DocID	**Document text**
1	hello
2	open house

Type B:

DocID	**Document text**
3	mi casa
4	hola Profesor

Type C:

DocID	**Document text**
5S	hola, y bienvenido
5E	hello, and welcome

The following is a translation of the Spanish words above for your own information. This glossary is NOT available to the IR system:

mi = my	
casa = house	
hola = hello	
profesor = professor	
y = and	
bienvenido = welcome	

a. Construct below the appropriate Term-Document matrix A to use for a corpus consisting of the above documents. For simplicity, use **raw frequencies**, rather than normalized TF.IDF weights. Make sure to clearly label the dimensions of your matrix.

A = **Ans.** We don't give the full matrix here, but the key point is that 5S and 5E must be merged into one column; unless there is at least some coccurrence of Spanish and English words in the same document vector (e.g., in the same column), LSI will be useless in doing cross-language information retrieval (CLIR).

The resulting term-document matrix has dimensionality 11 x 5. Entry (i,j) has the frequency of term i in document j (which in this case is always 0 or 1).

b. For the rank-2 approximation to A, given by $A_S = W_S S_S V_S^T$, Give the dimensions of the matrices A_S, W_S, S_S, and V_S^T. (A rank-2 approximation means that s = 2)

Ans. (The question really meant to ask "give the *useful*" dimensions of the matrices)
A_S is 11 x 5
W_S has a useful dimensionality of 11 x 2
S_S has a useful dimensionality of 2 x 2
V_S^T has a useful dimensionality of 2 x 5

c. State succinctly what the (i,j) entry in the matrix $A^T A$ represents.

Ans.
The (i,j) entry is exactly the # of words shared by doc i and doc j

d. State succinctly what the (i,j) entry in the matrix $A_S^T A_S$ represents, and why it differs from **c.**

Ans.
The (i,j) entry is the similarity of doc i and doc j in the 2-d LSI subspace. It is different from **c.** because of dimensionality reduction.