CS 276A : Text Information Retrieval, Mining, and Exploitation
Open Book Final Examination
Solutions
Monday, December 9, 2002

This final examination consists of 12 pages, 10 questions, and 80 points. 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 3 hours 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	Possible	Score
1 Short	10	
2 Link	10	
3 Dict	10	
4 Index	10	
5 Web	8	
6 XML	6	
7 ProbIR	4	
8 UI	8	
9 Rocchio	8	
10 Eval	6	
Total	80	

1. Short questions:

(a) Consider a bigram index for wildcard queries. Give an example of a sentence that falsely matches the wildcard query **mon*h** if the search were to simply use a conjunction of bigrams.

(b) Consider a web connected as a bowtie in which 20% of the pages are in IN, 60% in the SCC and 20% in OUT. [Hint: Refer to the Lecture 14 for a description of IN, SCC, OUT, and the bowtie structure of the web]. Suppose that you can initiate a crawl from 3 starting points, with the sole objective of reaching as many pages in this web as possible. Which of these would be best for starting the crawl from?

(i) One page each from IN, SCC and OUT.
(ii) One page from SCC and two from IN.
(iii) Two pages from SCC and one from IN.
(iv) Three pages from IN.

(c) Under the setup in (b) – if we were to pick three random pages from the web, which of (i-iv) is most likely?

Write 1 sentence of answer/explanation:

(d) How does stemming typically affect recall in Boolean document retrieval?

(e) Give three ways in which the standard TREC ad hoc IR task differs significantly from typical web search tasks

*Consider the following propositions, and give a brief (1 – 2 sentence) response. For each statement, we want to know if it is **true or false**, but also want an explanation as to why, and you will be graded mainly on the explanation.*

(f) Pseudo-feedback always increases precision and recall.

(g) It is easy to spam behavior-based ranking algorithms.

(h) The following two XML documents are different.

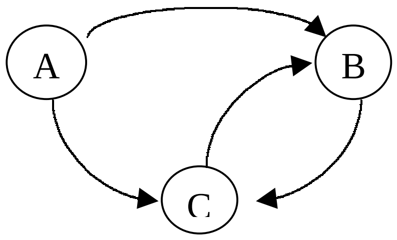
```
<recipe id="117" category="dessert">
  <title>Rhubarb Cobbler</title>
  <ingredients>
    <item><amount>2 1/2 cups</amount><type>diced rhubarb</type></item>
    <item><amount>2 tablespoons</amount><type>sugar</type></item>
  </ingredients>
</recipe>
```

```
<recipe id="117" category="dessert">
  <title>Rhubarb Cobbler</title>
  <ingredients>
    <item><amount>2 tablespoons</amount><type>sugar</type></item>
    <item><amount>2 1/2 cups</amount><type>diced rhubarb</type></item>
  </ingredients>
</recipe>
```

(i) An XML document guide will contain a particular element name like "title" only once.

(j) The structured document retrieval principle will rank the fragment highest that has the most occurrences of the query term. (Assume a one word query.)

2. Link analysis:
Consider a small web with 3 pages A, B and C. A links to B and C, while B links to C and C links to B. Compute pagerank, hub and authority scores for each of the three pages. Also give the relative ordering of the 3 nodes for each of these scores, indicating any ties.



Pagerank:
Assume that at each step of the pagerank random walk, we teleport to a random page with probability 0.1, with a uniform distribution over which particular page we teleport to.

Hubs/Authorities:
Normalize the hub scores so that the maximum hub score is 1.
Normalize the authority scores so that the maximum authority score is 1.

Hint 1: using symmetries to simplify and solving with linear equations might be easier than using iterative methods.

Hint 2: for partial credit, provide at least the relative ordering (indicating any ties) of the three nodes, for each of the three scoring measures.

3. Dictionary storage:
Consider a lexicon (dictionary) in which there are no words of length 1 or 2. Assume the probability of a word in this lexicon having length i is proportional to $1/i^2$, for $i > 2$. Assume that the distribution is truncated so that the longest possible word length is 25. Further, the lexicon has 50,000 words in it. [Hint: The sum of the series $1/i^2$ is $\pi^2/6$. If needed, you can approximate the sum of the series $1/i^2$ for i = 1, ... , 25 by 1.61.]

(a) Assume each character requires a byte of storage. Although inefficient, we could safely use 25 bytes per word for the dictionary, as in the first cut approach in class. How much storage is required for storing all the words in this fashion?

Now let us analyze a more efficient scheme.

(b) What is the expected number of distinct words (types) of length i as a function of i?

(c) From (b), infer the expected total number of characters to write down all words of length i as a function of i.

(d) Now consider storing the words as one contiguous string, with a term pointer to the beginning of each word. How much space is used in total, including storage for the entire string as well as for the pointers that resolve the beginning of each word? Show a formula, and estimate the total as a number of bytes needed.

(e) Next consider "blocking" the dictionary string so we point to the beginning of each eighth word rather than every word, storing the length of each word in one byte immediately preceding its appearance in the string. What is the total space used now? Show the formula, your estimates, and an estimated number of bytes.

4. Index construction

We have 100 million documents containing a total of 9 million terms.

(a) How many posting entries are there using the simple Zipf approximation used in class? You may assume that the natural log of 9 million is 16.

(b) Assume 12 bytes per postings entry and a machine with sufficient main memory to hold all data in memory. Assume a cost of 1 microsecond per cpu operation. How much time would it take to sort the postings entries in memory? Assume we use Quicksort with running time *2N* ln *N*.

(c) Next consider the case where we have only 8GB of main memory, and we use blocks with 200 million postings entries in a block. Assuming 0.5 milliseconds per disk seek, 0.5 microseconds per byte following a seek in block transfer mode and 1 microsecond for all other operations, estimate the time to create one such block of 200 million sorted postings on disk. Also give the total time needed to create all such (initial) sorted blocks.

(d) Now consider merging these sorted blocks into a single inverted index by reading two sorted blocks from disk, merging them and writing back to disk. What is the time for such a single such read/merge/write?

(e) What is the total merge time for all such merges in fully sorting the data?

5. Web size estimation:

(a) When estimating the size of the web using a random walk on web pages, we do not use the "teleport to a random web page" operation used in the pagerank computation. Why not?

(b) Consider the capture/recapture strategy for estimating the size of the web. We draw a random number of pages from one search engine and check whether they are indexed in a second search engine. Consider the test for whether the chosen page is present in another engine. List at least two sources of bias in this test.

(c) Two web search engines A and B each generate a large number of pages uniformly at random from their indexes. 30% of A's pages are present in B's index, while 50% of B's pages are present in A's index. What is the number of pages in A's index relative to B's?

(d) [**Note**: this question is nontrivial to answer. Attempt it only if you have time]
Now let us consider a scenario in which we use two crawls to estimate the frequency of duplicates on the web. Web search engines A and B each crawl a random subset of the web of the same size. Some of the pages crawled will be duplicates – exact textual copies of each other at different URLs. Assume that duplicates are distributed uniformly amongst the pages crawled by A and B. Further, for this exercise we will define a duplicate as a page that has exactly two copies. No pages have more than two copies. A indexes pages without duplicate elimination whereas B indexes only one copy of each duplicate page. [Note: the two random subsets have the same size *before* duplicate elimination.] If 45% of A's indexed **URLs** are present in B's index, while 50% of B's indexed **URLs** are present in A's index, what fraction of the web consists of pages that do **not** have a duplicate?

6. XML retrieval

(a) Retrieval T/F

Suppose we have a document collection where the documents are richly annotated XML documents (as might be found in a humanities or legal setting), such as the one shown partially in outline at right. Consider doing IR over such a collection where we could do (passage) retrieval over larger or smaller units of text (e.g., documents, sections, paragraphs, etc.). That is, we would choose some element as the indexing level, and return as matches units of that level. Because each term occurrence in the document is marked up with its part of speech, the "smallest" granularity is at the term level.

```
<document>
  <chap>
    <sec>
      <p>
        <s>
          <w t="PRP">I</w> <w t="VB">know</w> …
        </s>
        <s>
          <w t="NNP">Bill</w> <w t="VB">saw</w> …
        </s>
        …
      </p>
      <p>
      </p>
    </sec>
    <sec>
      …
    </sec>
    …
  </doc>
  <doc>
    …
  </chap>
  …
</document>
```

Consider the following propositions, and give a brief (1 – 2 sentence) response. For each statement, we want to know if it is **true or false**, but also want an explanation as to why. You will be graded mainly on the explanation.

Assume for concreteness that we are issuing a two-word query to the IR engine.

i. As the size of the indexing unit gets smaller the difference between using a Boolean term presence/absence model and a model using term frequency gets smaller and approaches zero.

ii. As the size of the indexing unit gets smaller the difference between using and not using document frequency scaling gets smaller and approaches zero.

iii. As the size of the indexing unit gets smaller, use of a binary “term-incidence vector” becomes an increasingly attractive way to index the text, in terms of space and time efficiency. [A “term-incidence vector” is also referred to as a "bitmap" and is the representation shown on slide 3–4 of lecture 1. The 0’s and 1’s are explicitly stored as bit vectors.]

iv. As the size of the indexing unit gets larger, the *relative* overhead in storage cost for doing a positional index vs. a simple frequency index increases.

(b) Xquery

For an XML collection in the format shown below, write an Xquery that retrieves all recipes that have an ingredient named "parsley". The XML collection is recipes.xml.

```
<?xml version="1.0" encoding="UTF-8"?>
<collection>
  <description>
     Some recipes used for the XML tutorial.
  </description>
  <recipe>
    <title>Beef Parmesan with Garlic Angel Hair Pasta</title>
    <ingredient name="beef cube steak" amount="1.5" unit="pound"/>
    ...
    <preparation>
      <step>
        Preheat oven to 350 degrees F (175 degrees C).
      </step>
      ...
    </preparation>
    <comment>
      Make the meat ahead of time, and refrigerate over night, the acid in the tomato sauce will tenderize
the meat even more. If you do this, save the mozzarella till the last minute.
    </comment>
    <nutrition calories="1167" fat="23" carbohydrates="45" protein="32"/>
  </recipe>
  ...
</collection>
```

7. Probabilistic language models for IR

We examined in class a probabilistic language model approach to IR based on a mixture model where the probabilities were based on the term frequency of a word in a document, and the collection frequency of the word. Doing this certainly assures that each word of a query (in the lexicon) has a non-zero chance of being generated by each document. But it has a more subtle but important effect of implementing a form of inverse collection frequency weighting. Explain how this works. In particular, include in your answer a concrete numeric example showing inverse collection frequency weighting at work.

8. User Interfaces

(a) The standard interface to the Stanford University Library (Socrates Basic Search) is a Form-based query specification, whereas the standard interface to most search engines (Google, Altavista, etc.) is a single search box. Is this choice just arbitrary/historical, or are there good reasons why these two different interfaces are suited to typical users and tasks? Give some specific examples in your answer.

(b) What is the difference between relevance feedback and query expansion?

(c) Why do commercial web search engines typically not provide relevance feedback functionality? Give at least 2 reasons.

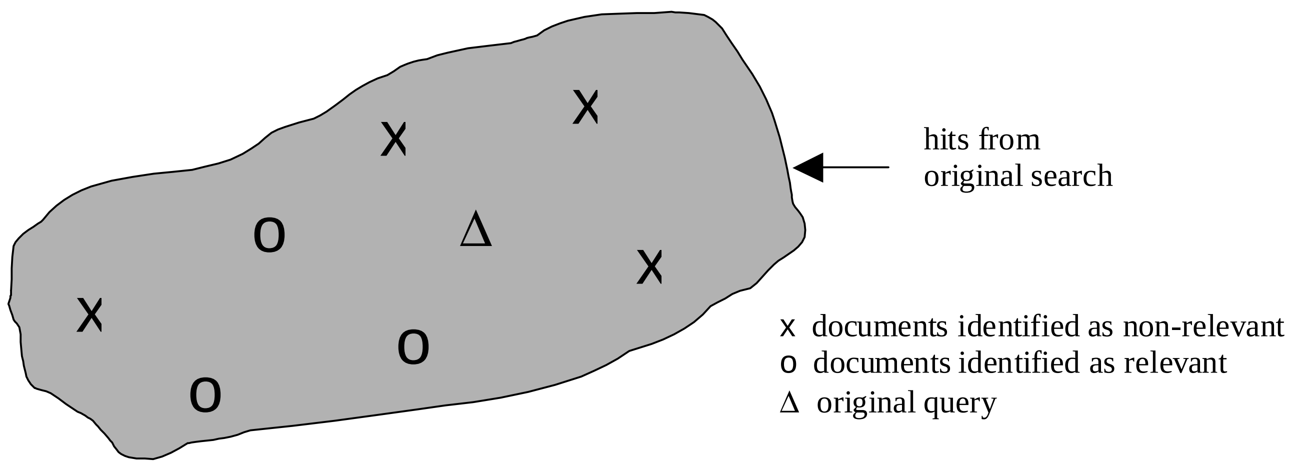
(d) Compare the following two techniques for query expansion: synonym expansion using a thesaurus, and automatic query expansion by common terms/phrases in top ranked documents. Describe advantages and disadvantages of each approach. (You should aim to provide at least 3 clear points of differentiation.)

9. Rocchio

If our document vectors are normalized, then they live on the unit hypersphere, and we can reasonably represent a small patch of the unit hypersphere as a flat surface (just as with maps in geography!). The below picture shows such a depiction of an initial query vector, the best hits, and whether they were judged relevant or irrelevant.

a. Consider now applying Rocchio' s algorithm. For fixed alpha = 10 and zero gamma, consider increasing beta. Draw a curve on the below figure showing the position of the reformulated query for different values of beta. Use the values beta = 0, 10, 100 and label those points on the curve. The points drawn don' t have to be in precisely the right place, but they need to be qualitatively correct. **Note: in the figure below, X refers to the non-relevant documents, O refers to the relevant documents, and Δ refers to the query.**

b. Suppose now that gamma is set to the same value as beta, where now will the reformulated query be for beta = gamma = 0, 10, 100?


X
X
O
Δ
X
X
O
O
hits from original search
x documents identified as non-relevant
o documents identified as relevant
Δ original query

c. One way to construct the Rocchio relevance feedback query is as the "optimal" separator between relevant and non-relevant documents. (Note that this is the theoretical optimal version of the Rocchio query and is different from the "practical" one assumed in a. and b.)

Optimality is defined as follows. Let $r_1,\ldots,r_m$ be the length-normalized vectors for the relevant documents, and $n_1,\ldots,n_k$ be the length-normalized vectors for the non-relevant documents. Then the Rocchio-optimal query ρ maximizes the difference between the average correlation of relevant and non-relevant documents:

$$\rho = \arg\max_{v,\|v\|=1}\left[\frac{1}{m}\sum_{r_i} v\cdot r_i - \frac{1}{k}\sum_{n_i} v\cdot n_i\right]$$

where *m* is the number of relevant and k is the number of non-relevant documents.

Given this definition how is ρ computed? Show your work and justify your final answer.

10. Evaluation

For this exercise, we define the precision-recall graph of a result list as the set of (precision/recall) points, where one precision/recall point is computed for each returned document. We will initially define the breakeven point as the point where precision equals recall.

a. Can there be more than one breakeven point? Explain.

b. Some precision recall graphs do not have a breakeven point as defined above. An alternative definition is: The breakeven point is the point with the smallest difference between precision and recall among all those that have larger precision than recall. Write a rough pseudocode program that computes the breakeven point (you should not need more than a few lines)

Answers

Note: due to time constraints, the solutions are not in a "polished" format. Also, there were often times other accepted answers presuming the approximations / assumptions were valid.

1. Short answer

note: for T/F, the explanation is what mattered

a) A string such as **"Monkey Breath"** has all the necessary bigrams. Some students neglected the 'h$' bigram. Other students tried to stuff all the necessary bigrams into one word, which is OK, but there was no reason to do so ("**moonh**" was a popular correct answer, although I've never heard of that word).

b) (iv) -- 3 pages from IN could get you a few more pages. Everything in SCC and OUT is reachable from IN/SCC.

c) (iii) Clearly the answer is either (i) or (iii), so compute Pr[] for them. Pr[(i)] = 6 * .2 * .6 * .2 = .144, and Pr[(iii)] = 3 * .6 * .6 * .2 = .216, so the right answer is (iii). Most people got it right but their actual probabilities were wrong (you have to acct for unordered permutation). We didn't penalize for this.

d) Stemming generally increases recall, because you will find not only words with the form of the word you entered, like "persuade" but also related forms like "persuasion". For boolean queries with negation, recall might be lowered.

e) Trec Ad Hoc vs. web: Long queries vs. short queries. Topic queries vs. commonly queries for a specific named entity. Precision over the recall range is important vs. precision on top 10 documents being almost all that matters. Not able to use hypertext vs. able to use hypertext. Etc.

f) False. It depends on whether the first k documents returned were actually relevant as assumed by pseudo feedback.

g) True. Just write a robot that issues a query and clicks on the document you want to get boosted.

h) True. Order of elements matters.

i) False. If there are different paths leading to elements with the same name (e.g., document title vs. section title), then the corresponding name will occur more than once.

j) False. In general, a leaf fragment will have fewer occurrences of the term, but be more relevant.

2. Link Analysis

Power method works, but the below is simpler and takes a only few lines of work.

Pagerank:

Since the indegree of A is 0, the rank of A is .1 * 1 / 3 = .033 (from teleport)
By symmetry and uniqueness of the solution, B = C
By definition of pagerank, B = .9 (1/2 A + C) + .1 * 1/3
So by simple algebra
A = .033
B = C = 0.4785

Hubs and Authorities:

Authority:

There is no concept of teleport, so A = 0
B = C
So A = 0, B = C = 1

Hub:

Use the Authority scores above, and iterate once to get
A = 2, B = 1, C = 1
Normalized, A = 1, B = 1/2, C = 1/2

3. Dictionary Storage
Let N=50,000
a. 25*N*
b. Normalizing constant C = 1 / [1.61 – **(1 + 0.25)**] = 2.8. N = 50,000. Formula is: C *N* / i^2 for i > 2
c. *iCN* / i^2 = CN / i
d. Assume a w byte pointer for each word: Nw + sum_i=3,25 [quantity from c]. Assume w = 4 and that the sum of 1/i for i = 3...25 is about ln 25 – **(1 + 0.5)** = 1.72. Then we have N(4 + 1.72C) = N(4 + 4.82) = 8.82N = 440K. \ceil{log_2 4.82*N} = 18 bits = 2.25 bytes is a better choice for w, in which case the answer is N(2.25 + 4.82) = 353K (t sum_i=3,25 1/I = 2.32 if you don't approximate 2.8*2.32 = 6.5 N*(2.25 + 6.5) = 731K)

e. There are now Nw/8 bytes for the pointers. And there are now [sum_i=3,25 CN / i] + D bytes for the actual words and lengths. So we now have N(w/8 + 4.82) + D ≈ 6.3N = 315K (or with w = 2.25, we get 305K). (again, needs t o be changed if the above sum is not approximated (2.32)

4. Index Construction
These have not been double checked yet
4a. number of docs * log(number of terms) = 100 10^6 * log(9 10^6) = 100 10^6 * 16 = 1.6 10^9
4b. 2 n log n = 2 * 1.6 10^9 log(1.6 10^9) = 2 * 1.6 10^9 * 21.2 = 4.24 10^10 = 4.24 10^4 seconds
4c. number of blocks: 1.6 10^9/ 2 10^8 = 8 ; sort time: 2 n log n = 2 200 10^6 l(200 10^6) = 7.64 10^9 ;
read/write time 2*0.5 * 10^3 + 2*12*0.5*200 10^6 = 2400 10^6 = 2.4 10^9 ; total time: 8*(2.4+7.64) 10^9
4d. merge: (number of ops)*200 10^6 where number of ops should be 13-15 (6 words read,6 words stored, 1-3 comparisons), so <3 10^9. read/write: (number of ops in block transfer mode)*12*2*200 10^6 = 2*2*200 10^6 = 4.8 10^9 ; 4 disk seeks = 2 10^3
4e. log(2,8) = 3; 3 * 4 * (time to merge one block)
(safely ignore disk seeks)

5. Web Size Estimation
a. Pagerank is run on a set of known web pages that are already crawled. We can teleport since we can randomly select pages from this set. [Estimating the size of this subset of the web is trivial: just count.] The point of web size estimation is that we don't already have an exhaustive list of the pages of interest. So we can't teleport.
b. 1) We assume that the population search engines draw from is the whole web. It is not: for example, non-connected, non-submitted pages are not part of the population. 2) We assume that each page has the same probability to be drawn. That is not correct: popular/important pages (yahoo.com) are more likely to be drawn than obscure pages. 3) duplicates
c. |A| = 5/3 |B|

d. (This question was not originally designed to be this subtle, hence the disclaimer)
The size of A is pN if p is the sampling rate and N the size of the web. Let 1-2f be the proportion of non-duplicates.
Let B be the crawl with dups.
Let B' be the dup free version of B.

Deterministically, we know
(1) |A| = |B| = pN
(2) |A^B' | = .45 |A| = .45 |B| = .50 |B' |
so that
(3) |B' | / |B|= 0.90

We expect:
(4) We expect that |A^B' | = p * 0.9 |B| = p * 0.9 * pN = 0.9p^2N
Using (1) & (2), we have that 0.9p^2N = 0.45 pN, so that
(5) p = 0.5

We now have p, but still need to find f:

We expect:
B to have 2fp^2N dups with both copies in B, so that fp^2N pages get dropped. But using (3), the number dropped must be 0.10|B|, so that fp^2N = 0.10|B| = 0.1pN, so that f = 0.1/p = 0.1 / 0.5 = 0.2

So f = 0.2. 1-2*f = 60% of the web has no duplicates.

6. XML Retrieval
a.
- **(i)** True. The limiting case of small is indexing at the word level, and then every word is absent or has tf of 1, and there is no difference, but even for indexing at the sentence level, it is unlikely that a (non-stopword) term will occur more than once, so tf' s are still essentially binary.
- **(ii)** False. In the limiting case of word indexing, document frequency would still have the effect of causing the low df word "documents" to be returned before the high df word "documents". At a slightly larger level like sentences, it would have its usual effect of giving more preference to documents with cases of the high df word in the ranking.
- **(iii)** False. As the indexing unit gets smaller, a bitmap gets an increasingly space inefficient way to index, since it doesn' t exploit the sparsity of present vocabulary items in each document. It would make it relatively slower too.
- **(iv)** Comparing the limits, indexing at the word level means that there is no additional cost to adding a positional index, because every word is at position 1. Indeed, what you have is equivalent to indexing at the collection level and maintaining everything in the positional information. Therefore, assuming equivalent quality compression of both parts, the sum of the sizes of the document postings and position postings should be roughly constant. But the size of the document postings shrinks as the indexing unit gets larger, so the overhead of having a positional index grows greater. . [We may want to be more specific about what type of positional index is meant here. A postings list with position information I assume? Maybe it doesn't matter.]

b.

```
FOR $r IN document("recipes.xml")//recipe
WHERE $r//ingredient[@name="parsley"]
return $r
```

7. Probabilistic Language models

Suppose we are searching with the query "rare common" where "rare" is a rare word (its cf is 10 in the 10,000,000 word document collection) and "common" has cf 10,000. Now suppose we have two documents of length 1000 words: d_1 in which "rare" appears once, but not "common", and d_2 which has "common" but not "rare". And assume that the mixture weight between the two models is 0.5 for simplicity.

Then P("rare common"| d_1) = 0.5*(0.001+0.000001)* 0.5*(0+0.001) = 2.5025 x 10^{-7}
P("rare common"| d_2) = 0.5*(0+0.000001)*0.5*(0.001+0.001) = 5 x 10^{-10}
So, d_1 with the rare word ranks much higher than d_1 with the common word.

In general, the thing to note is that even though the cf component of the mixture model is constant over all documents, the effect of it is that for a high cf word, presence or absence of the word in the document will have relatively little effect on the probability assigned by the model (e.g., factor of 2 in the example above), whereas for a low cf word, presence or absence of the word in the document can have a very large effect (factor of almost 1000 in the example)

8. UI

a. A library catalog has clear fields, which are largely uniform over the entire collection (author, title, year of publication, publisher, etc.) and which are the main things people want to search on. While web search has some clear possible fields (e.g., language restriction) for most standard searches, they do not belong to a small number of clear slots, and conversely there would appear to be less value in restricting a search to some slot.

b. In relevance feedback the user directly or indirectly judges documents as good or bad, and the document vectors modify the query. In query expansion, users directly or indirectly provide information on whether additional search terms or good or bad.

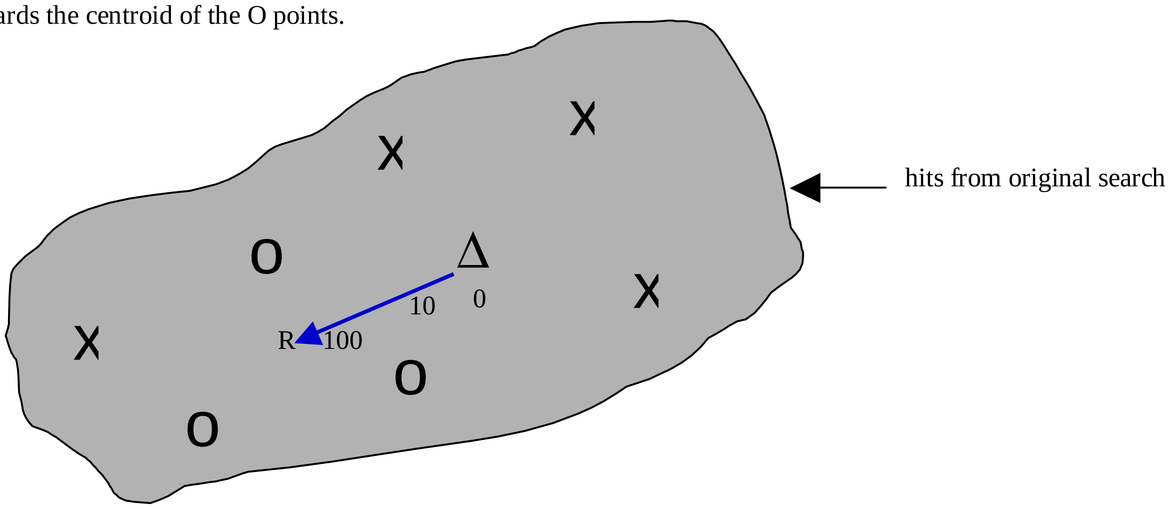
c. Relevance feedback requires an extended search session, whereas most web users never look beyond the first page of results, let alone considering query refinement. Relevance feedback is apparently too complicated for most of the great unwashed to understand and use. The not-very-sparse query vectors that result from relevance feedback are expensive to run in a typical IR engine, and so use of relevance feedback goes against being able to service many queries cheaply.

d. Thesaurus synonym expansion is fast at runtime, while query expansion based on top match is more expensive/slower. Thesaurus synonym expansion requires a priori available domain-specific resources whereas query expansion based on top documents doesn' t. Automatic query expansion is sensitive to particular language patterns present in the corpus, whereas thesaurus-based expansion isn' t. A thesaurus can be easily tuned by a human to achieve particular results, whereas automatic query expansion cannot. Automatic expansion is more likely to hurt precision than thesaurus synonym expansion.

9. Rocchio

a. See the figure.

b. The centroid of the X points is roughly where the original query result lay. Therefore, for gamma = 0, we get the same result as before, and for gamma = 10, 100 we have faster movement along the blue line towards the centroid of the O points.



X documents identified as non-relevant
O documents identified as relevant
Δ original query
R reformulated query

c.

rho = argmax_[v,length(v)=1] v [1/R sum_(ri) ri - 1/N sum(ni) ni]

The unit vector with the largest inner product with a vector v is v/||v||.

So if rho' = 1/R sum_(ri) ri - 1/N sum(ni) ni
then rho = rho' / ||rho' ||

10. Evaluation

10a.

Consider a result list with two breakeven points at ranks n1 and n2. Let the total number of relevant documents be R. Let the number of relevant documents in the first n1 and n2 documents be r1 and r2.

The definition of breakeven point is:

r1/n1 = r1/R
r2/n2 = r2/R

Solutions are r1=0,r2=0,r1=r2.

So there is only one non-zero breakeven point. There can be additional breakeven points with precision=recall=zero.

We gave full credit for one of the following two solutions 1) an example with two breakeven points (one being zero) 2) the above proof that there can only be one non-zero breakeven point.

10b.

Go down ranked list until you find first point where recall<precision. Go back one.
[boundary case: there are no points with precision>recall]