

CMPSCI 646, Information Retrieval (Fall 2002)

Midterm exam, solutions

With the exception of Problem 1, all parts of every problem were weighted equally. For Problem 1, part (a) was 16 points, part (b) was 10 points, part (c) was 5 points, and part (d) was three points.

Problem 1. Compression and inverted lists (30 points)

- (a) For each of the following areas within an IR system, indicate how the area can benefit by compression and why there is possible benefit. Briefly state and support the criteria for choosing a compression algorithm for each of those areas (it is perfectly plausible that you select the same algorithm for each area).

The areas are: (1) the original text, (2) the inverted lists, and (3) the ranked result lists.

Solution: *There are lots of things that could have been said for this problem, here are the main points.*

(1) For the original text, any decent compression algorithm would do. However, it should be possible to access a single document easily, so something that allows random access into the text is important.

(2) For the inverted list, the expected answer was something that justified delta encoding, and then either delta or gamma coding. Other algorithms are plausible, provided they provide for fast decompression.

(3) A meaningful answer for a result list depended on what you thought a ranked list contained. It's likely to include a score, a unique document identifier, and maybe a title. If the score is a floating point number, most compression algorithms we talked about in class won't work well: you could convert it to an integer. If there is text, than almost any text algorithm would work. No random access is needed here.

- (b) Slide 19 (page 10) of the lecture on indexing includes a sketch of an algorithm for using inverted lists to rank documents in a vector space model. That algorithm is a term-at-a-time algorithm. It could also be written as a document-at-a-time algorithm that would compute the score for document 1, then document 2, then document 3, and so on. Assuming inverted lists are stored sorted by document number, sketch the resulting document-at-a-time algorithm. Remember that inverted lists are sparse, so they are represented as (d_i, w_i) pairs where d_i is a document number and w_i is the weight of the term in that document.

Solution: *This problem was a mistake. There's really not much point to it in a vector space model (the problem should have talked about a Boolean model.) Most people presented an algorithm like,*

```
for each document
  for each query term
    fetch the term's inverted list
    find the document's entry
    update the scores
  end
end
```

Impressively inefficient, but it does allow the skip lists to serve a purpose for part (c): they can be used to find the document's entry more quickly.

A better solution would pull in the start of all inverted lists for the terms in the query and then slowly walk through them. If the query were an AND query, then you could rapidly cycle through some of the documents for some of the lists. Alas, you weren't asked to consider an AND query....

- (c) Skip lists (mentioned on slide 22 of the same lecture) provide a mechanism for skipping large portions of the inverted list for some queries. Modify your algorithm of part (b) to incorporate skip lists for efficiency.

Solution: *See the discussion of part (b).*

- (d) How might the presence of skip lists affect your choice of compression algorithm for an inverted list? That is, would you answer part (a.2) differently and if so how?

Solution: *Skip lists create jumps to random parts of the inverted list—e.g., skipping directly to document 10534. That means that the entries cannot be compressed using any algorithm that doesn't allow the scanning to start at arbitrary points. For example, delta encoding makes it difficult to jump into a list, because the value depends on all values before it. This issue is easy to fix by providing some sort of synchronization either as part of or in addition to the skip list. But something has to be done.*

Problem 2. Distributed IR (20 points)

A distributed IR system sends a query to a number of search engines, accepts ranked lists back from each of them, and then merges the results. If the systems are all based on the vector space model, their scores will typically include some sort of IDF component.

- (a) Assuming each of the systems is identical (i.e., only the collections they index are different), why is it not a good idea to just merge the ranked lists based on the systems' returned scores?

Solution: *IDF is a heuristic designed to discount the impact of very frequent words. Suppose we organize databases by subject and issue a query concerning subject A. The document frequency of words in our query will be much higher for collection discussing subject A, and consequently IDF scores will be lower. This will lower the scores for all documents coming from collection A, compared to documents that come from collections with incidental mention of A. This is highly undesirable as we will end up penalizing the "good" collections and miss a lot of relevant documents.*

- (b) Language modeling systems do not include an IDF component. Assuming that every system is a language modeling system implementing the query likelihood model, would it be reasonable to just merge their scores? Assume probabilities are estimated using maximum likelihood of the document smoothed by the local collection's statistics.

Solution: *No. A language modeling system ranks documents D by the following formula:*

$$P(Q|D) = \prod_{w \in Q} \left[\lambda \frac{tf_{w,D}}{|D|} + (1 - \lambda) \frac{cf_w}{|C|} \right]$$

The collection frequencies cf_w will be higher for the collection discussing the subject of the queries, so the overall document scores will also be higher. This seems like a good idea – we will favor documents from the right collection. However, the approach will boost the scores for all documents in the subject collection, both relevant and non-relevant. It will also reduce the variance of the scores: their sensitivity to whether or not the query word actually occurs in a given document. We will end up ranking a lot of non-relevant documents from the subject collection above the potentially relevant documents from other collections.

Problem 3. Probability Ranking Principle (30 points)

The year is 1977 and you serve as editor-in-chief of the Journal of Documentation. You receive a paper by S. E. Robertson, in which he states that he came up with the best possible way to rank documents in a given collection. He claims that optimal performance can be achieved if you rank documents D by the decreasing ratio of probabilities: $\frac{P(D|\mathcal{R})}{P(D|\mathcal{N})}$. Here $P(D|\mathcal{R})$ is the probability of observing document D in the relevant class \mathcal{R} , and correspondingly $P(D|\mathcal{N})$ is the probability of observing D in the non-relevant class \mathcal{N} . Before you allow the paper to be published, you need to determine whether Robertson's wild ideas are correct.

You will prove that ranking the documents D by the ratio $\frac{P(D|\mathcal{R})}{P(D|\mathcal{N})}$ will lead to highest expected average precision.

- (a) Let C be a collection of N documents, and let D_1, D_2, \dots, D_N be some ranking of these documents. Some of these documents are relevant, but we don't know which ones. Let $P(D_i \in \mathcal{R})$ be the probability that document D_i is relevant. For a given rank r , let $Rel(1, r)$ denote the number of relevant documents in ranks $1 \dots r$. Prove that we can compute the expected number of relevant documents in ranks $1 \dots r$ as follows:

$$E\{Rel(1, r)\} = \sum_{i=1}^r P(D_i \in \mathcal{R})$$

Solution: Let $Rel(i) = 1$ if $D_i \in \mathcal{R}$ and $Rel(i) = 0$ otherwise. Then $Rel(1, r) = \sum_{i=1}^r Rel(i)$, and:

$$E\{Rel(1, r)\} = E\left\{\sum_{i=1}^r Rel(i)\right\} = \sum_{i=1}^r E\{Rel(i)\} = \sum_{i=1}^r (0 * P(D_i \notin \mathcal{R}) + 1 * P(D_i \in \mathcal{R})) = \sum_{i=1}^r P(D_i \in \mathcal{R})$$

- (b) Using part (a) provide an expression for expected precision at rank r in terms of probabilities $P(D_i \in \mathcal{R})$.

Solution: $prec(r) = \frac{Rel(1, r)}{r}$, and consequently

$$E\{prec(r)\} = \frac{1}{r} E\{Rel(1, r)\} = \frac{1}{r} \sum_{i=1}^r P(D_i \in \mathcal{R})$$

- (c) Prove that for every r , expected precision at rank r is maximized when the documents D_i are ranked in order of decreasing $P(D_i \in \mathcal{R})$.

Solution: Proof by contradiction. Let $D_1 \dots D_N$ be an optimal ranking, i.e. for every r , expected precision at rank r is maximized. Suppose this ranking does not follow $P(D_i \in \mathcal{R})$. Then for some rank r , we can find two other ranks $a < r$ and $b > r$ such that $P(D_a \in \mathcal{R}) < P(D_b \in \mathcal{R})$. According to (b), $E\{prec(r)\}$ includes $P(D_a \in \mathcal{R})$, but does not include $P(D_b \in \mathcal{R})$. If we swap D_a and D_b in the ranking we can increase expected precision, since $P(D_a \in \mathcal{R}) < P(D_b \in \mathcal{R})$. This is a contradiction, since we assumed that $D_1 \dots D_N$ was an optimal ranking.

- (d) Use the definition of average precision and result of part (c) to show that average precision is also maximized if we rank documents D_i in order of decreasing probabilities $P(D_i \in \mathcal{R})$.

Solution: Let $m_1 \dots m_k$ denote the ranks where relevant documents occur in the overall ranking. Then expected value of average precision is:

$$E\{avg.pr\} = \frac{1}{k} \sum_{j=1}^k E\{prec(m_j)\}$$

Part (c) implies that ranking by $P(D_i \in \mathcal{R})$ leads to highest possible $E\{prec(m_j)\}$ for each rank m_j . Therefore the sum of $E\{prec(m_j)\}$ is also maximized.

- (e) In parts (a) - (d) we used the notation $P(D \in \mathcal{R})$ to denote the probability that document D would be judged relevant. A more conventional notation for this probability is $P(\mathcal{R}|D)$. Use Bayes' rule to show that ranking documents by the ratio $\frac{P(D|\mathcal{R})}{P(D|\mathcal{N})}$ is equivalent to ranking by the posterior probability $P(\mathcal{R}|D)$. Conclude that ranking by the probability ratio (probability ranking principle) leads to maximum expected average precision.

Solution:

$$P(\mathcal{R}|D) = \frac{P(D|\mathcal{R})P(\mathcal{R})}{P(D|\mathcal{R})P(\mathcal{R}) + P(D|\mathcal{N})(1 - P(\mathcal{R}))} = \frac{1}{1 + \frac{1-P(\mathcal{R})}{P(\mathcal{R})} \frac{P(D|\mathcal{N})}{P(D|\mathcal{R})}} = f\left(\frac{P(D|\mathcal{N})}{P(D|\mathcal{R})}\right)$$

Where $f(x) = (1 + cx)^{-1}$, and c denotes the ratio $\frac{1-P(\mathcal{R})}{P(\mathcal{R})}$, which is constant and independent of D . It is easy to show that f is monotonically decreasing, thus ranking by $\frac{P(D|\mathcal{R})}{P(D|\mathcal{N})}$ is equivalent to ranking by $f\left(\frac{P(D|\mathcal{N})}{P(D|\mathcal{R})}\right)$.

- (f) We have just demonstrated an optimal ranking principle, which is guaranteed to produce maximum precision at every level of recall. Why doesn't this solve all problems in IR?

Solution: In order to use the principle we have to be able to compute $P(D|\mathcal{R})$, and $P(D|\mathcal{N})$. PRP does not specify a way of computing these probabilities. In the IR literature it has become common practice to assume independence of the words in a document and compute $P(D|\mathcal{R})$ and $P(D|\mathcal{N})$ by multiplying word probabilities. While this sounds reasonable it actually leads to sub-optimal rankings even if we were able to estimate $P(\cdot|\mathcal{R})$ and $P(\cdot|\mathcal{N})$ very accurately. In addition, while it is fairly straightforward to estimate $P(\cdot|\mathcal{N})$, it is almost impossible to estimate $P(\cdot|\mathcal{R})$ since no examples of relevant documents are provided in a typical retrieval situation.

Problem 4. Relevance feedback (20 points)

The slide on the bottom of page 9 of the relevance feedback notes talks about a system that modifies the document representation as part of relevance feedback. (Typically relevance feedback is viewed as modifying the query to bring it closer to relevant documents.) Let $D_J = \{(Q_1; J_1); (Q_2; J_2); \dots; (Q_{D_J}; J_{D_J})\}$ be a set of judgment pairs for document D , where J_i is either 1 (for relevant) or 0 (for not relevant).

- (a) Assume that documents are represented as vectors \vec{D} in a vector space model. Show a sketch of the algorithm used to modify the representation of document D in response to judgments D_J .

Solution: Let $R = \{Q_i : J_i = 1\}$, be the set of queries to which D was judged relevant, and $N = \{Q_i : J_i = 0\}$ be the set of non-relevant queries. Let D_w be the weight of word w in the original vector representation of D . Let Q_w be the weight of w in the vector representation of some query Q . The weight of w in the updated representation would be:

$$D'_w = \alpha D_w + \frac{\beta}{|R|} \sum_{Q \in R} Q_w - \frac{\gamma}{|N|} \sum_{Q \in N} N_w$$

The values of parameters α , β and γ can be found empirically.

- (b) Describe how such an approach might be used in the language modeling approach, assuming the query-likelihood model. That is, given a set of past judgments D_J , how might the probabilities be estimated differently?

Solution: As before, let $R = \{Q_i : J_i = 1\}$, and $N = \{Q_i : J_i = 0\}$. The idea is similar to part (a), but we have to be careful: we are dealing with probability estimates, which must be positive and have to sum to one. Suppose $P(w|D)$ is the probability initially assigned to word w . A possible update equation is:

$$f_w = \frac{\alpha}{|R|} \sum_{Q \in R} \frac{tf_{w,Q}}{|Q|} - \frac{\beta}{|N|} \sum_{Q \in N} \frac{tf_{w,Q}}{|Q|}$$

$$P'(w|D) = \gamma P(w|D) + (1 - \gamma) \frac{\max(f_w, 0)}{\sum_v \max(f_v, 0)}$$

Where α , β and γ are between 0 and 1, and can be tuned for optimal performance.