CMPSCI 646, Information Retrieval (Fall 2003)
Midterm exam solutions

Problem CO (compression)

1. The problem of text classification can be described as follows. Given a set of classes, $\mathcal{C} = \{C_i\}$, where each class contains documents, assign a new document D to the class C_i that is the best match for it. Generally "best match" means that they discuss the same topic or are otherwise have strongly related content. You can assume that each class has numerous training instances, $D_{i,j} \in C_i$ for $j = 1, \ldots, |C_i|$.

   One approach to classification first builds a compression model for each class. When a new document, D, is considered for classification, it is compressed with respect to *each* class and is assigned to the one that yields the most compression. Why does that work?

   (10 points) The key point is that a document will compress well if it uses terms in a way that is not surprising with respect to a compression model. So if frequent terms in the document are likely in the model, then the document will compress well.

2. Inverted lists are normally sorted in ascending order of document identifier. There are some situations where it instead makes sense to store them sorted by the descending weight of the term in the document—i.e., the first document listed will have the highest weight for that term, the second will have the next highest, and so on. What, if any, are the implications for inverted list compression of using that scheme instead? Support your claim.

   (10 points) In this situation it is unlikely that delta encoding (finding the "gap") will be useful since the document IDs will now be somewhat random in the list. That means that they will stay large and will not compress as well.

   Note that it is plausible that the weights will now be compressible, depending on how they are represented. For example, if the weight were just an integer, then they would be monotonically decreasing and so a gap could be used there.

   Also note that term position information within documents could still be compressed as before since that would not have changed. Note all indexes include term position information.

3. LSI can be seen as a compression scheme because it reduces the number of dimensions in the vector space. Is LSI a "lossy" or a "lossless" compression scheme? Explain what it means to be whichever you choose, and support your claim.

   LSI is lossy. After decomposition, some number of dimensions is selected and the less significant values are discarded. That means it is no longer possible to reconstruct fine distinctions: only the key "concepts" are retained.

Problem LM (language models)

For these problems it will be helpful to have a copy of Ponte and Croft's SIGIR 1998 paper:

PDF	`http://ciir.cs.umass.edu/pubfiles/ir-120.pdf`
PostScript	`http://ciir.cs.umass.edu/pubfiles/ir-120.ps`

Note that because of some font issues, the PDF version is not very legible on the screen, but it should print fine.

1. Is the Ponte and Croft a query likelihood model, document likelihood model, a combination, or something else?

   Query likelihood since it is estimating the probability of the query's being generated.

2. The probability estimator they provide is,

$$\hat{p}(t|M_d) = \begin{cases} p_{ml}(t,d)^{(1.0-\hat{R}_{t,d})} \times p_{avg}(t)^{\hat{R}_{t,d}} & \text{if } tf_{t,d} > 0 \\ \frac{cf_t}{cs} & \text{otherwise} \end{cases} \quad (1)$$

Although that is a fine probability estimator for the Ponte and Croft model (a Bernoulli model), it is not appropriate for the multinomial model that dominates language modeling in IR. Why isn't $\hat{p}$ appropriate then?

The (multiple) Bernoulli and Multinomial models have different event spaces. In both models, there is one event X_t per vocabulary term t. In the Bernoulli model, X_t is either 1 or 0, i.e. X_t measures the presence or absence of a term in a given document d. The estimator $\hat{p}(t|M_d)$ gives the probability that the term t is present in document d. These events X_t are considered independently, so $\sum_t \hat{p}(t|M_d)$ does not have to be 1 (instead, the sum of the probability that the term does and the probability that it does not occur must be one). Intuitively, if a document talks about computer networks*, both terms would have a probability of occurrence close to 1 (with sum of probabilities > 1).*

In the multinomial model, the events X_t are counts of terms in d and hence are not independent. Here $\sum_t \hat{p}(t|M_d)$ must be 1, so the term probabilities from the Ponte & Croft paper are not valid for the multinomial model.

3. Regardless of how the probabilities are estimated, the final function that is calculated for each document is,

$$\hat{p}(Q|M_d) = \prod_{t \in Q} \hat{p}(t|M_d) \times \prod_{t \notin Q} (1.0 - \hat{p}(t|M_d)) \quad (2)$$

This is a multiple Bernoulli model rather than a multinomial model that is more commonly used in IR.

   (a) How does this model take term absence (in the query) into account and what difference might it make? Illustrate with an example.

   In this model, a query is expected to include all of the high probability terms in a document. So a document will match a query less well if it includes words that are not mentioned in the query. The multinomial model does not take that into account.

   (b) In what way does this model fail to take repetitions (in the query) into account and what difference might that make? Illustrate with an example.

   The Bernoulli model is only concerned with term presence (or absence), not with how often the term occurs. So have a word in a query 100 times is no different than including it twice.

4. A major disadvantage of Equation 2 is that it is a product over all terms in the vocabulary, not just over terms in the query. At first glance, this seems to mean that inverted files will not provide any efficiency. Describe some pre-calculating and manipulation that you could do to implement this efficiently. *Hint: A = A*B/B or, if you use logs, A = A - B + B.*

For each document, calculate $X = \prod_{t \in V}(1.0 - \hat{p}(t|M_d))$, where V is the entire vocabulary (not just words in the query). Then, noting that $V = \{t \in Q\} \cup \{t \notin Q\}$, we can calculate:

$$\begin{aligned} \hat{p}(Q|M_d) &= \prod_{t \in Q} \hat{p}(t|M_d) \times \frac{X}{\prod_{t \in Q}(1.0 - \hat{p}(t|M_d))} \\ &= \prod_{t \in Q} \frac{\hat{p}(t|M_d)}{(1.0 - \hat{p}(t|M_d))} \times X \end{aligned}$$

Note that X can be done at indexing time so is a constant at query time. And now the calculation involves only words in the query and is much faster.

5. Why do most language modeling implementations calculate, store, and manipulate the logarithms of probabilities?

The main reason is to avoid floating point underflow from multiplying very small numbers together. Adding logarithms is also faster than multiplying.

6. What does it mean to say that a language model is *generative*? Is the vector space model generative? Why or why not?

   A generative model explicitly includes the capability of generating members of the set of documents—i.e., can generate text. The vector space model does not incorporate any information about how text or documents are constructed: it merely says that if you convert text to vectors, then distance between vectors corresponds to relevance. The vector space model is not generative.

Problem EV (evaluation)

1. In the class project, several queries were generated and run to find top-ranked documents for each topic. A pool for judging a particular topic was developed by selecting the documents at rank 1 for every submitting system, then those at rank 2, 3, and so on, until there were at least 120 unique documents found (recall that the sets of top-ranking documents probably overlap).

   Those 120 documents were then sorted by document identifier, divided into six sets (top 20, next 20, ..., and bottom 20 in alphabetical order), and handed out to six people to be judged (20 documents each). Suppose that an individual fails to return his or her 20 documents, forcing us to assume that they are all non-relevant.

   What will be the impact on the evaluation measures for that topic? How significant is the impact? Is any system likely to be more impacted than another? Focus on recall, precision, and average precision, though you may discuss other measures if you like.

   (8 points). The first thing to note is that the sorting of the documents by document identifier may have an impact on the evaluation measures. It is likely that there is a correlation between document identifiers and relevance to specific queries. For example, the query "mars" will likely return `nasa-`$*$ *documents that are possibly even consecutively numbered. That way, it is possible that many relevant documents are assigned to a single person. If all of these documents are assumed to be non-relevant, there would be a significant impact on the evaluation measures.*

   Precision: The mislabeling of relevant documents can cause some precision values to be smaller than they should be. The higher the rank of the mislabeled documents, the greater the impact.

   Recall: The average effect on recall is probably not very big. There will be fewer points where recall increases (compared to complete relevance judgments), but the total number of relevant documents has also decreased. Recall at specific ranks can go up or down, depending on the rank of the mislabeled documents.

   Average Precision: Since the precision values tend to be smaller, it is likely that average precision will go down (compared to the measure with complete judgments). However, there are also situations possible where average precision increases.

   In general, better systems tend to be more affected by the mislabeling of relevant documents, since they retrieve more relevant documents. If a system retrieves only few (or no) relevant documents, its evaluation scores will only be slightly affected.

2. Even if all 120 documents are judged, there are still hundreds of documents that were not judged for that topic. What impact does *that* have on evaluation scores? Again, focus on recall, precision, and average precision.

   (6 points) Here we'll contrast the true evaluation measures with those calculated based on the assumption that all non-judged documents are non-relevant. If more relevance judgments were available, the recall values at higher ranks would likely be lower, precision values at lower ranks would be higher. Average precision would likely be lower, since there are probably more points where recall increases and the precision at those points tends to be low.

3. Mean Reciprocal Rank (MRR) is an evaluation measure used for evaluating systems when there is precisely one relevant document. Each topic is given the score $1/r$ where r is the rank at which the relevant document was found. In this situation, what are the pros and cons of MRR as opposed to mean average precision?

   (5 points) If there is exactly one relevant document, Mean Reciprocal Rank is equivalent to Mean Average Precision.

4. The lecture notes present one method for interpolating precision values for recall values between measured recall points. The approach is to take the maximum precision value "to the right" of the recall point in question. Specifically:

$$P(R) = \max\{P' \quad : \quad R' \geq R \wedge (R', P') \in S\}$$

It would also be possible to interpolate by drawing straight lines between the measured points. This was called "connect the dots" in the "revised" evaluation lecture notes. Argue either against or for "connect the dots" in contrast to the "max to the right" interpolation described above.

We'll argue for "max-to-the-right". That interpolation technique forces the recall-precision graph to follow the intuition that precision drops with increasing recall, whereas "connect-the-dots" exhibits a saw-shape pattern. In addition, the "connect-the-dots" technique gives a non-function, which can have infinitely many precision values for a particular recall point.

Problem RF (relevance feedback)

1. Why might "pseudo-relevance feedback" sometimes transform an imperfect query into one that is incredibly bad?

   An imperfect query will retrieve mostly non-relevant documents at the top of the ranked list. PRF will "assume" they are relevant and modify the query to retrieve more documents like those. The result will probably be a query that retrieves almost entirely non-relevant documents.

2. Suppose a query had precisely one relevant document in a collection. Speculate on whether or not "local feedback" (of any type) would be likely to help. Support your speculation.

   If the relevant document were retrieved at rank one, it couldn't be improved, so no type of LF (or PRF) would help.

   If the relevant document were not retrieved in the top ranked documents at all, then PRF would be more likely to move the query toward non-relevant documents than toward the missing relevant one. So it wouldn't help.

   If the relevant document were in the top ranked set, PRF techniques would incorporate it into their calculations, but it would be dominated by the larger number of non-relevant documents. It is unlikely that it would help in that case, either.

3. In a sense, LSI builds a global association thesaurus that implicitly causes query expansion (except that it's done at indexing time). In what ways is the "expansion" caused by LSI different from that caused by "local feedback" at query time? What impact is that likely to have on effectiveness?

   The key difference is the local feedback finds term associates using top-ranked documents that are similar to the query, whereas LSI uses all documents in the collection. If a term has multiple meanings, LSI will probably find them all and not be able to distinguish them. If the query contains other words that disambiguate that word, then the associates in the top ranked documents are less likely to be ambiguous. So local feedback will work better.

4. Consider the following equations that describe how the Local Context Analysis approach weights candidate expansion concepts with respect to a query (the same equations appear in the lecture notes):

$$\begin{aligned} f(c,Q) &= \prod_{w_i \in Q} (0.01 + \text{co\_degree}(c, w_i))^{idf(w_i)} \\ \text{co\_degree}(c,w) &= \max\left(\frac{n_{cw} - En(c,w) - 1}{n_c}, 0\right) \\ En(c,w) &= \frac{n_w n_c}{N} \\ idf(w) &= \min(1.0, \log_{10}(N/n_w)/5) \end{aligned}$$

What is the purpose of the min() operator on the idf function? (The lecture notes indicate it is to "cap the IDF component", but that's an error.) In answering this question, you should indicate how often a term must occur in order to have its IDF component be higher than one, and why it might not make sense for it to become smaller than one.

The min() operator keeps the IDF value from going above 1. It would otherwise be above one for any term that occurs in less than 1/100000 of the documents in the collection. That is, for extremely *rare terms.*

Note that co-degree is between 0 and 1. If IDF is greater than 0 but less than 1, then using it as a power will make the co-degree larger. So the IDF value is boosting the co-degree weight of lower IDF terms: the less common the term (higher the IDF), the less the co-degree term is boosted. If IDF were to go above one, it would start dropping the weight of the feature—if IDF got large enough, it would force the co-degree term toward zero. To prevent that reversal, the IDF value is not permitted to rise above one.