CS630 Lecture 10: Relevance Feedback Methods
Date: February 28th, 2006
Lecturer: Lillian Lee
Scribes: Ari Rabkin and Victoria Krafft

In this lecture, we'll examine the idea of applying relevance feedback (RF) to the three models we've studies so far; classic probabilistic retrieval, the vector space model, and the language model.

For now, let (RF) = explicit binary judgments on each of the top k retrieved documents. This means we don't have to deal with a lack of feedback, or documents classified as "maybe relevant".

Contents

1 Relevance Feedback in Classic Probabilistic Model

We'll start with the case of probabilistic retrieval, because this case is the clearest. This is actually not the order of historical development; query expansion RF methods for the VSM are significantly older. Recall the RSJ "weight" for term $v^{(j)}$ assuming binary attributes ,

$$RSJ_j = \frac{P(A_j = 1|R = y)}{P(A_j = 1)} \cdot \frac{P(A_j = 0)}{(P(A_j = 0|R = y)}$$

The parts which depend on $R = y$ use the "positive" feedback from RF (documents judged relevant) to re-estimate the probability a document is relevant.

What about negative feedback here? We can argue that it's in the model indirectly. Note that in the odds ratio version of RSJ, the $P(A_j = 1)$ is actually denoted $P(A_j = 1|R = n)$. In practice, the general collection is used to estimate this value, rather than the collection of not relevant documents. The argument for this is that general collection is so much larger, and overwhelmingly not relevant, that it's more meaningful to use that than to use the handful of known-to-be-irrelevant documents.

However, when we display the top-k documents, negative feedback is interesting. These are documents our search engine put in the top k documents, and the fact that they're not relevant seems like it ought to be useful. That said, they don't get used in most RF schemes for probabilistic retrieval. It's possible that we may not normally see significant numbers of documents in the top k which are not relevant, so these don't occur frequently enough to be useful.

The general scheme for RF is as follows:

- Initially: Present some set of documents, using the Croft & Harper estimate for $P(A_j = 1|R = y)$, or using q as the relevant document.
- Get RF.
- Re-rank with new RF information.
- We repeat this until the user throws stuff at us, or goes away, potentially satisfied.

How effective is this in scenarios of interest to us? For example, suppose there is some d = "autos", q = "cars", and only relevant documents contain "autos". Eventually, you want the score for d to be high enough that it is retrieved. What is the score for the document? Recall that RSJ score for d given query q is:

$$\prod_{j:q_j>0, a_j(d)>0} RSJ_j$$

And this only uses terms which appear in d and q, so we'll never look at "autos". No matter what we do, this query simply isn't going to get us documents whose words are not contained in the query. One solution: add indicative terms to the queries. This is called *automatic query expansion* (abbreviated AQE).

Here's a scheme proposed by Robertson in 1990: rank terms by

$$\text{selectscore}(v^{(j)}) = RSJ_j \times [P(A_j = 1|R = y) - P(A_j = 1)]$$

This is known in the literature as the wpq score for reasons that will become clear in the next paragraph. We add the terms with the highest selectscore rank to the query, then re-score the documents using the RSJ score, updated to take RF into account. Robertson presents a mathematical derivation of this approach.

Isn't the second term in the selectscore redundant with respect to RSJ_j? Let's suppose we have two terms, v and v', with corresponding attribute variables A and A' (we are avoiding superscripts and subscripts for notational clarity). Let

$$p = P(A = 1|R = y),\ q = P(A = 1)$$

Likewise, let

$$p' = P(A' = 1|R = y),\ q' = P(A' = 1)$$

Suppose that

$$RSJ = \frac{p/(1-p)}{q/(1-q)} < RSJ' = \frac{p'/(1-p')}{q'/(1-q')} \tag{1}$$

so that v' is preferred by RSJ_j ranking. Is it possible to simultaneously have the selectscore values

$$RSJ \cdot (p-q) > RSJ' \cdot (p'-q') \tag{2}$$

Equation (1) implies: $\frac{p}{q} < \frac{p'}{q'}$, and hence equation (2) implies $p - q > p' - q'$. This is saying that the relative difference between the probability of a given term appearing in a relevant document and that term appearing in a general document is bigger for v than for v', but the absolute difference is smaller. For example, consider $p = \frac{1}{2}$ and $q = \frac{1}{4}$, while $p' = \frac{7}{100}$ and $q' = \frac{1}{100}$. In this case, the rankings differ because they treat the difference between the two things differently: the RSJ weight selects only for relative difference in term frequency between relevant and not relevant documents, *selectscore* for the absolute difference also. A large absolute difference "requires" that the term appear frequently which means selection is biased towards more frequent terms; moreover small changes in the absolute frequency between relevant and not relevant documents do not result in words being put into the query.

It turns out that automatic query expansion is significantly older than this, and indeed, it was used in VSM systems as long ago as 1971.

2 Relevance Feedback in the VSM

[Rocchio '71, Ide & Salton '71]

Consider the vectors shown in Figure 1. The query vector $\vec{q}$ is closer to $\vec{d'}$, which is not relevant, than to d, which is relevant, which means that the wrong ranking will result (assuming a cosine retreival function, say). If we add $\vec{q}$ to $\vec{d}$, then we get a new vector $\vec{q'}$, which is closer to the vector for the relevant document d. We can also take $\vec{q}+\vec{d}$ and subtract $\vec{d'}$, giving us $\vec{q''} = \vec{q} + \vec{d} - \vec{d'}$, which is further away from documents which are not relevant, and closer to documents which are relevant. Note that $\vec{q''}$ could have negative components. This would mean that we would actively discourage the system from handing us documents containing the corresponding terms. This is nice, since it lets you get some of the power of boolean queries, but negative terms are commonly zeroed in practice.

The Rocchio formula is

$$\vec{q_{new}} = \alpha\vec{q} + \beta\frac{1}{|R|}\sum_{d^{(i)}\in R}\vec{d^{(i)}} - \gamma\frac{1}{|NR|}\sum_{d^{(i)}\in NR}\vec{d^{(i)}}$$

where R is the set of documents judged relevant, and NR is the set of documents judged not relevant. Taking the average of the [ir-]relevant documents reduces class-size bias effects. Otherwise, if we got a large number of (nearly) identical documents, and marked them all as relevant, that might swamp other information (like the original query!). The constants α, β, and γ are free parameters, where large α (relative to β and γ) gives more weight to the initial query, while large β and γ weight those documents that have been marked relevant/not relevant more. γ is sometimes set to 0, since feedback about irrelevant documents may not always be reliable.

In practice, despite the pleasing simplicity of adding vectors to make a new query vector, there's often some term filtering. For instance, one might only allow a term to enter into the new query if it appears in half the relevant documents.

3 Relevance Feedback for LM-based IR

We had two derivations of the LM model. The first derivation produced the scoring function $P(Q = q|D = d, R = y)P(R = y|D = d)$. If we use a Bayes flip on the $P(R = y|D = d)$ bit, then you can use RF directly– modulo the problem with the first quantity being relevance for

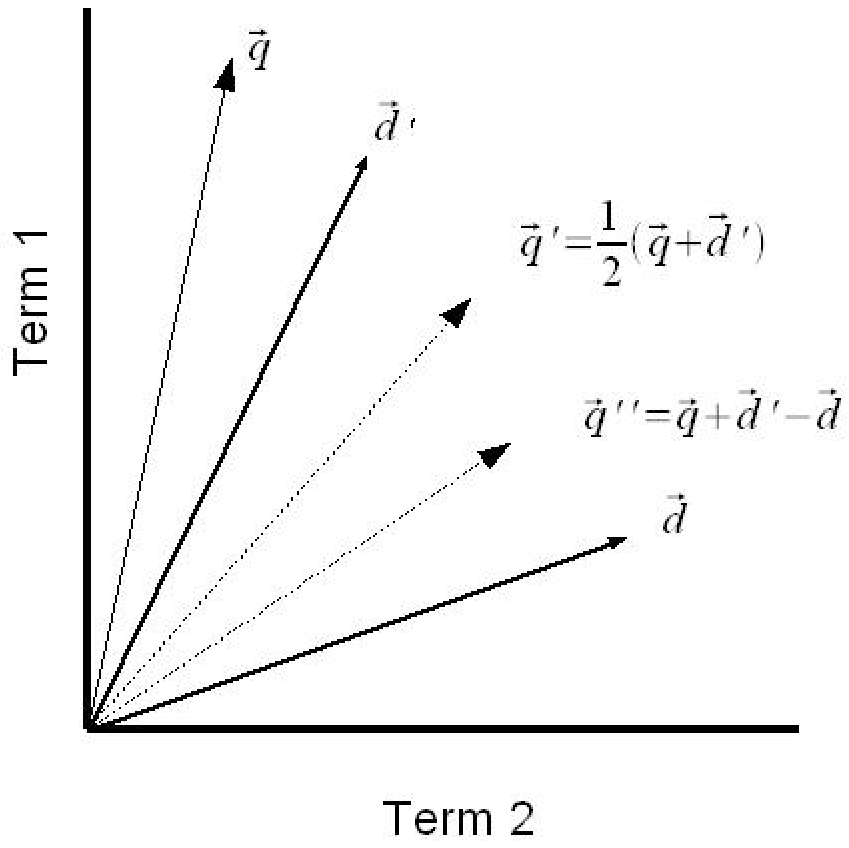
$\vec{q}$
$\vec{d}\,'$
$\vec{q}\,'=\frac{1}{2}(\vec{q}+\vec{d}\,')$
$\vec{q}\,''=\vec{q}+\vec{d}\,'-\vec{d}$
$\vec{d}$
Term 1
Term 2

Figure 1: A sample vector space, with query q and documents d and d'. Document d is relevant, d' is not.

the specific document in question. The second derivation started from $[R = y] \equiv [T_d = T_Q]$. Here, the relevance does not exist as a distinct random variable. What could be done instead is to use the RF to update the estimate of T_Q. This is more sensible if we use the distance between T_Q and T_D, instead of requiring an absolute match for relevance to be "declared".

The inability to really use relevance information directly is one of the common attacks made against the LM approach to IR. There is a variant of LM (the Relevance Model, proposed in Lavrenko & Croft '01) which tries to bridge the gap between LM and the probabilistic models.

4 Conclusions

This has been an overview of the use of RF in the three retrieval paradigms we have studied. Note that all these techniques have been query specific: the feedback is probably only useful for this particular query from this particular user, which is a serious limiting factor (exceptions include obvious and common mis-spellings). Even then, queries could have multiple meanings. We're also asking people to do a lot of work to give us the feedback.

One last point: automatic query expansion can be very disorienting to users. If they don't see the terms that the AQE adds, they can be very confused why documents that don't match the query are being returned. It's very frustrating for a user to get back documents they didn't ask for, and don't know how to un-ask for.

For example, suppose a user enters "Lilian Lee" (note the typo) in google. The user will get the homepage for "Lillian Lee". This is rather annoying; users are getting the homepage for a different person than they searched for. This isn't exactly due to AQE, but it has a similar result: pages being returned that don't match the query the user entered.

5 Question

a) Consider the following proposed RF mechanism: Given a set of relevant documents, find their centroid in some VSM. Rank the documents in the set by distance from this centroid. How does this approach compare to the approach we discussed for the VSM?

   Answer: This is a special case of the VSM with RF which we discussed in class. You can get this mechanism by using the Rocchio model and setting $\alpha = \gamma = 0$, and $\beta = 1$. This choice of parameters means that the new query is:

$$\frac{1}{|R|} \sum_{d^{(i)} \in R} \vec{d}^{(i)}$$

   This is exactly the centroid of the set R.

b) What happens in Robertson's scheme for AQE if you have a term t that is negatively correlated with relevance?

   Answer: Nothing (as long as a reasonable number of terms are more positively correlated with relevance); if the term was in the query it will stay there, otherwise it will not be added. If the term is negatively correlated with relevance, then $P(A_j = 1|R = y) < P(A_j = 1)$, and therefore $P(A_j = 1|R = y) - P(A_j = 1) < 0$ This means that term j will always be ranked at the bottom by *selectscore*, and so the term won't be added to the query unless no terms are positively associated. That case is probably pathological, since it implies that the relevant documents have no terminology in common that isn't already in the query.

c) In part b, we asked about terms that are negatively correlated with relevance. Explain how AQE could be used to incorporate negatively correlated terms into the probabilistic model. What problems could arise from trying to consider these terms? What benefits would they offer?

   Answer: This is a very broad topic, and the answers provided below are only some of the many responses.

   One way to add these terms into the model is to find the terms with sufficiently negative selectscore. Then, add those terms to the query when computing the RSJ score for the query. Those terms will have a very small $P(A_j = a_j(d)|R = y)$, and a large $P(A_j = 0|R = y)$, so the rank of a document containing negatively correlated terms will be reduced.

   If we take negatively correlated terms into account, we make it possible for a user to get rid of documents on topics that are not relevant to the query. This effectively provides a form of boolean "NOT" operator, allowing for disambiguation between synonyms, such as "John Smith", the actor in a TV show, and "John Smith", the famous computer scientist via the weighting of other terms in the updated query.

   Finding terms that are negatively correlated may be tricky. The documents you will be examining are the documents in the top k and there may not be many negatively correlated terms in those.

   If a term is incorrectly marked as being negatively correlated, then the user may not have a chance to correct this, since documents with that term will not be displayed. In contrast, if a term is incorrectly marked as being positively correlated, the user will get documents with this term to evaluate.

   This scheme has the peculiar feature that it assigns nonzero rank to documents which only contain the query terms with negative relevance correlation; if we hadn't added these terms to the query, the documents wouldn't be retreived at all. This could probably be fixed by some sort of post hoc processing, but in practice these documents will have low ranks, and probably will never be seen.