CS630 Lecture 15: Preference IF
Date: March 28th, 2006
Lecturer: Lillian Lee
Scribes: Ari Rabkin and Victoria Krafft

Today we'll finish up on Preference Implicit Feedback, and then discuss the midterms. The relevant papers are: Joachims '02, Radlinski & Joachims '05. (Both Joachims and Radlinksi are in the Cornell CS department)

Contents

1 Using Comparative Relevance **1**

2 Exam Discussion **3**

3 Questions **4**

1 Using Comparative Relevance

As you may recall from the last lecture, we discussed how to use clickthrough data to get relative relevance judgments about two documents in a pair.

For example ['02]: Suppose that for some query q, Google gives summaries s_i and s_j, with s_i closer to top of page. If s_i not clicked on, and s_j is clicked on, then s_i is probably less relevant than s_j. By only looking at cases where s_i is closer to the top of the page, we alleviate the "preference" bias towards links towards the top of page.

This concept can also be extended to query chains ['05]. Suppose for q_k, we have summaries $s_1 \dots s_n$, and the user doesn't click on any of them. The user then reformulates the query to q_{k+1}, and clicks on s'_i. One inference we can make is that s'_i more relevant than $s_1 \dots s_n$. However, there is a potential bias: eye-tracking data shows that s_1 and s_2 might be the only summaries that were actually read. Because of this, we probably should not make inferences about $s_3 \dots s_n$. That said, we can safely assume that $s'_i > s_1$ and $s'_i > s_2$, with respect to q_k. Studies showed that this inference was highly accurate.

Let's talk about reference feedback in general for a moment. There are some significant problems with all the RF methods we have seen so far.

Problem 1: The RF methods we've seen so far can't use preference information. The RF methods we've discussed use the feedback to compute quantities like $P(A_j > 0|R = y)$. A preference doesn't really tell us whether or not to consider a document relevant.

Problem 2: The RF may not generalize to other queries. The feedback has been obtained for a particular query from a particular user.

Suppose we consider a vector space incorporating query features explicitly. Instead of representing d as $\vec{d}$, represent (q, d) as $\phi(q, d)$. Note that the features of (q, d) need not simply be features of q and d separately, but can include features based only on the combination. (More technically, (q, d) need not be a tensor product of vectors q and d)

For example, $\phi_i(q, d)$ could be the cosine between "standard" $\vec{q}$ and $\vec{d}$, which we used before.

Alternatively, we can associate specific terms with specific documents. We might have:

$$\phi_i(q,d) = \begin{cases} 1 & \text{if } d = \text{CU's homepage and } q \text{ contains "big red"} \\ 0 & \text{otherwise} \end{cases}$$

Or, we could use information other than the document's content:

$$\phi_i(q,d) = \begin{cases} 1 & \text{if } d \text{ is from a finnish site (.fi)} \\ 0 & \text{otherwise} \end{cases}$$

We can even use "metasearch" information:

$$\phi_i(q,d) = \begin{cases} 1 & \text{if } d \text{ is ranked first by Google for } q \\ 0 & \text{otherwise} \end{cases}$$

As you've probably gathered, these $\phi(q, d)$ vectors live in a very high-dimensional space, but most of their entries are probably zero. These are highly sparse vectors.

How do we use this? We're going to try to produce a vector $\vec{w}$ with the property that inner products with respect to $\vec{w}$ honor preference orderings. (The name $\vec{w}$ is supposed to indicate that this vector specifies the weight accorded to each attribute.) That is, if d was judged by the user as more relevant than d' with respect to q, then we should have $\vec{w} \cdot \phi(q, d) > \vec{w} \cdot \phi(q, d')$

This is essentially building a support vector machine using the user's relevance feedback as the training data.

Computing such a $\vec{w}$ is unfortunately an NP complete problem. Instead, we relax our requirement to finding a $\vec{w}$ so that the inner products with it violate our preference constraints as little as possible (subject to some additional regularization constraints). This is feasible, and in fact SVM construction is a fairly standard technique in AI and machine learning. The really clever thing about this work is that it reduces RF information retrieval to a known learning problem, where query dependence has been folded into the data to be classified. This really "works": the SVM technique can outperform Google.

How practical is all this? Ranking newly seen documents and queries is straightforward: find the vectors $\phi(q, d)$, compute their inner products with $\vec{w}$, and rank. The attributes that

don't depend on q can be computed ahead of time and stored, of course. We could also only apply this technique to some subset of the corpus, perhaps restricting to the top 100 results from some other search system.

2 Exam Discussion

At this point, we're going to break off, and quickly go over the problems on the midterm. These are sketches of important points, not complete solutions.

1) The question asked what would happen if we replaced the Croft-Harper estimate with the estimate that $P(A_j = 1|R = y) = \frac{N(j)+\alpha}{N+\alpha}$ Recall that the Croft-Harper estimate from lecture assumed that this was a constant for all terms in query.

1c) Recall that the RSJ model uses: $\frac{P(A_j=a_j(d)|R=y)}{P(A_j=a_j(d))} \cdot \frac{P(A_j=0)}{P(A_j=0|R=y)}$. Since $P(A_j = 0|R = y) = \frac{N-N(j)}{N+\alpha}$, and $P(A_j = 1) = \frac{N(j)}{N}$, this reduces to $1 + \frac{\alpha}{N(j)}$, which is an IDF. This is the same qualitative kind of result as C-H, so our more "accurate" estimate didn't really change the result.

2) Rel: d generated by t_1 or t_2?
$P(R = y|D = d) = \sum_{t=1}^{n} P(T_d = t, R = y, D = d) = \sum_{i=1}^{2} P(T_D = t_i|D = d) = \sum_{i=1}^{2} \frac{P(D=d|T_D=t_i)P(T_D=t_i)}{P(D=d)}$. Using a uniform distribution to model $P(T_d = t_i)$ is a legitimate approach.

2b) This cancels out $P(D = d)$. This is a real boon, because it's not clear how to estimate $P(D = d)$.

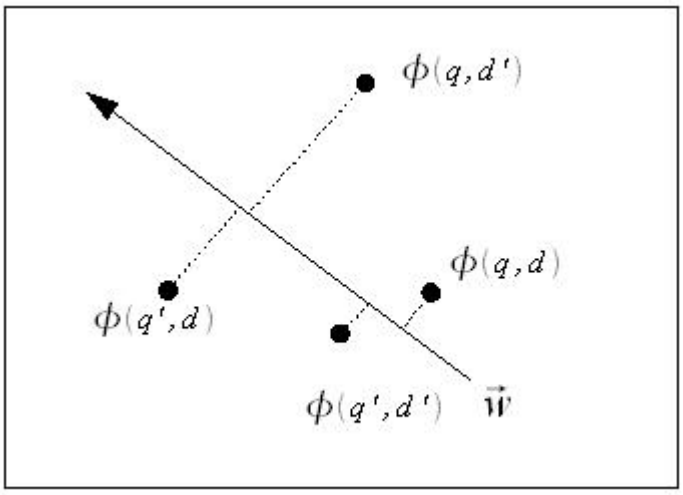

Figure 1: A sample vector space, with queries q and q' and documents d and d'. Document d is more relevant to q', and d' is more relevant to q. Dashed lines show orthogonal projections.

3) This one was just algebra; it's the trick we've seen before of breaking up the product.

4a) The motivation for inserting this term was that we wanted something that was monotonically increasing to a limit as tf_j grows, since we're using it to approximate the result of the 2-Poisson model, which had such a property, but is unpleasantly complex.

4b) The VSM one was: $\frac{tf_j(d)}{\sqrt{\sum_{k \neq j} tf_k(d)^2 + tf_j(d)^2}}$ This actually has the limit behavior that was used to motivate the Okapi/BM25 scoring function.

5) Documents with every term the same number of times are penalized heavily, in that they shrink substantially. Vectors on the axes (where only one term appears) don't shrink at all. There's no good reason to think this behavior would improve performance.

3 Questions

a) The Joachims model lets us choose any attributes we like, and uses the preference data to determine which attributes are important. If we made a poor initial choice of attributes, $\vec{w}$ could have inconveniently many dimensions. Which attributes do we drop? (Or equivalently, how do we reduce the dimension of the space?)

Ans: Drop the ones in which $\vec{w}$ has the smallest component, since it means that those attributes do not contribute significantly to the inner product with $\vec{w}$ unless the test data exhibits very large variation in such attributes, and hence do not affect the ranking much. This is essentially projecting $\vec{w}$ in such a way as to maximize its length in projection).

b) Can you find a $\vec{w}$ and a set of attributes so that the scheme reproduces cosine normalized tf-idf ranking?

Ans: The simplest answer would be to take $\vec{w}$ as the one-dimensional unit vector (1), and $\phi(q, d)$ to be the tf-idf weighted $\cos(\angle(\vec{q}, \vec{d}))$
There's also a slightly less trivial answer. Suppose we take our attributes to be the set of all terms in the corpus, and take $w^{(j)} = 1/N^{(j)}$ (where $N^{(j)}$ is the number of documents containing term $v^{(j)}$). We then choose $\phi_j(q, d) = fractf_j(d) \cdot tf_j(qnorm(d)$.

c The projections of $\vec{w}$ into various subspaces of the attribute space contain useful information. Give some examples of meaningful projections.
Ans: The length of the projection of $\vec{w}$ into the subspace corresponding to a given attribute is proportional to the attribute's value as a discriminator between relevant and irrelevant documents.
Alternatively suppose we take $\vec{w'}$ to be the projection of $\vec{w}$ into the subspace consisting of all query-independent attributes. Then the positions of the documents along $\vec{w'}$ correspond to their relevance independent of particular queries–their "generic" relevance or usefulness, as it were.