CS630 Lecture 13: Implicit Feedback

Lecture by Lillian Lee
Scribed by Randy Au, Nick Gerner, Blazej Kot

March 9, 2006

In this lecture, we outline an implicit feedback mechanism described by Shen, Tan and Zhai in their 2005 paper. It is a language modeling approach - it aims to build an information need language model P^*

Implicit feedback search is an iterative process, with the system displaying results and the user clicking on them, and then re-formulating the query in turn. At each iteration k, the data used to create an information need model P_k^* is as follows:

- The current query q_k
- The query history consisting of all past queries $qh(k) = q_1 \dots q_{k-1}$
- The clickthru history $ct(k) = ct_1 \dots ct_{k-1}$ where ct_i is the concatenation of summaries that the user clicked on in the ith iteration of the process.

1 Ideas

1.1 First idea: representing history via language models

First, let's represent all data sources as language models over 1-term sequences. For the query history, one approach is to simply average language models built from individual previous queries, this looks like:

$$P_{qh(k)}(v) = \frac{1}{k-1} \sum_i P_{q_i}(v)$$

The question then arises, is this a valid language model over 1-term sequences? We should check this before we go further.

If we assume:

$$\sum_v P_{q_i}(v) = 1$$

Then:

$$\sum_v P_{qh(k)}(v) = \frac{1}{k-1} \times (k-1) = 1$$

And similarly for the clickthru model $P_{ct(k)}$. Notice that at this stage, all the members of the histories have equal weight. Later we will see how to "age" the data, so more recent queries are more significant than older ones.

Note, incidentally, that this averaging of LM's to produce a new LM is reminiscent of averaging document vectors to create a new document vector in the VSM. Indeed, we can think of a document language model as being a vector similar to the VSM:

$$d \leftrightarrow (p_d(v`1)), p_d(v`2)), p_d(v`3)), ...)^T$$

1.2 Second idea: Interpolate between Current Query and History

Now let's think about how to get a ranking from our language models. We can define our information need model as a combination of the individual language models as follows:

$$\mathrm{P}^*_\mathrm{k}(v) = \alpha P_{q_k}(v) + (1-\alpha)(\beta P_{qh(k)}(v) + (1-\beta)P_{ct(k)}(v))$$

where $\alpha, \beta \in [0, 1]$.

Note that this looks similar to the Rocchio model for relevance feedback (using only positive feedback) in the VSM. However, this differs from VSM scoring which is roughly the sum of the product of term frequencies between the query and the document being scored (with some additional normalization, etc.). With language models, we need to consider the probability distributions induced by the query and interaction history, $P^*_k(v)$, and documents $P_d(v)$. For this, we can use something like the Kullback-Leibler divergence:

$$\sum P^*_k(v) \log \frac{P^*_k(v)}{P_d(v)}$$

This gives the expected log likelihood ratio with respect to P^*_k between the probability assigned to a term by P^*_k and the probability assigned to that term by P_d, which makes sense when considering a user's information need. Furthermore, there's a nice entropic interpretation as follows:

$$D(P^*_k||P_d) = \sum_v P^*_k(v) \log P^*_k(v) - \sum_v P^*_k(v) \log P_d(v)$$

The first term is the negative entropy of P^*_k which can eb thought of as the inherent uncertainty of knowing whether $V = v$ if $V \sim P^*_k(v)$. This is constant over all documents and can be dropped under ranking which yields the following ranking function:

$$-\sum_v P^*_k(v) \log P_d(v)$$

Ignoring the negative log function, this does end up looking a lot like the VSM scoring function for the Rocchio approach.

1.3 Third idea: Choosing a "Principled" Value for α

We need to answer the questions of when to trust the current query versus the history. This is really asking how to choose values for α and β in a principled way.

One choice is to give longer queries bigger α values. This can be implemented as follows.

We treat q_k as observed data, and the history as priors:

$$P_k^*(v) = \frac{tf_j(q_k) + \mu P_(qh(k))(v) + \lambda P_(ct(k))(v)}{|q| + \mu + \lambda}$$

This is Dirichlet style smoothing which we've seen before. This gives:

$$\alpha = \frac{|q|}{|q| + \mu + \lambda}$$

and β as a function of μ and λ by derivation.

1.4 Fourth idea: Aging History Data

We want to "age" old history items, so that they are less significant. This assumes that more recent data is more useful. A simple way to do this uses q_k as observed and $P^*_{(k-1)}(v)$ as a prior:

$$P_k^*(v) \propto tf_j(q_k) + \mu P_{k-1}^*(v)$$

for μ less than one. This uses previous models in the current model by some weighting value μ. Expanding this out shows that all previous $P^*_{(k-l)}(v)$ contribute to the current $P_k^*(v)$, but with each contributing less than the more current ones. This is only a simple choice using a single parameter (for simplicity). We could have used a set of μ_i values.

2 Evaluation

We want to evaluate based on computing $P_k^*(v)$ and $P_q^*(v)$ at each iteration k of the query process. Recall that evaluation of feedback systems had posed some problems to us in the case of explicit feedback systems. However, we can do our comparison using standard evaluation techniques because the user isn't giving explicit feedback to the system. But we do need hard(er) tasks to ensure many (or at least several) iterations. Specifically, Shen, Tan, and Zhai used 30 known "hard" TREC tasks giving hard queries and relevance data without clickthrough histories. Three judges generated clickthrough histories and query alterations over four iterations (3 query reformations). Shen, Tan, and Zhai found that their system did well on the third and fourth iteration (The second didn't provide enough information to do any accurate inference). Also, query aging was found to improve performace, while clickthrough aging was found to hurt performance. Also, clickthrough data was

found to be very important (almost to the excusion of prior queries), even though only 30% of the ducments corresponding to the clicked summaries were relevant

3 Questions

3.1 Question:

Shen, Tan, and Zhai performed an evaluation in the "standard way", that is to say, they ran their system and considered the performance at each iteration and compared it to the previous iteration and compared the performance of their system (using history information) against a system which only used the query provided. We argued that this is "ok" in the sense that there's no additional bias introduced as there was no explicit relevance information provided to the system. How valid is this argument in general? Specifically: how can you show that the argument is invalid (and discuss how the setting we have above differs from the explicit feedback settings we considered before)? What are (some) characteristics of settings in which this argument would be invalid?

3.2 Answer:

The argument is invalid if a system can somehow use the implicit relevance data to "unfairly" boost its performance by flooding its results with unprincipled but known-to-be-correct documents. In the explicit feedback settings we considered before, this is a severe problem because relevance is determined by the user who is also providing explicit (relevant or not) ratings for each document at each iteration. An unscrupulous system might provide every document in the system (or some subset) to the user and in the end only return those documents selected by the user.

Here, the user only provides implicit feedback and the system has no way of knowing that the clicked document is relevant or not. However, this is corpus specific. We learned that in the corpus and query load used in the above evaluation that only 30% of clicked documents were relevant. If this number were greater (or even at this low value) it might be enough for an unscrupulous system to boost its measured performance unfairly. For example, supposed the setting were one of simple question answering (as a short summary) and some background information (Supplementary to the user's information need) via a clickthrough. A system could simply take all the clicks as relevant responses in this case.