CS 630 Lecture 13: 03/09/2006
Lecturer: Lillian Lee Scribes: Peter Babinski, David Lin

An LM-based Approach Using Implicit Feedback

I. Shen, Tan & Zhai (2005) Paper
- **a. Building an Information-Need Model**
- **b. 1st Idea: Represent Data as (Non-smoothed) LM's over 1-term Sequences**
- **c. 2nd Idea: Use Weighted Average**
- **d. 3rd Idea: "Principled" Values for α**
- **e. 4th Idea: Aging**
- **f. Results**

I. Shen, Tan & Zhai (2005) Paper

a. Building an Information-Need Model

The paper by Shen, Tan, and Zhai (2005) describes a concrete example of a language model approach to relevance feedback. The basis for their approach is a construction of an information-need model. We will define this model as P_k^*. The star represents the fact that our model will use several different sources of information – the current query q_k, the query history, $qh(k) = (q_1,...,q_k)$, and the clickthrough data history, $ch(k) = (ct_1,...,ct_k)$. Let ct_i be the concatenation of summaries clicked on in the ith round. For each term v, $P_k^*(v)$ is the probability of term v if a one-word "query" were generated according to our model of the user's information need in the kth round based upon our history information.

b. 1st idea: Represent Data as (Non-smoothed) LM's over 1-term Sequences

First, we will try and represent the query and click-through histories using language models over 1-term sequences. The simplest way to represent the query in this fashion is to take the average over all LM's from the history data. For the query history:

(1) $$P_{qh(k)}(v) = \frac{1}{k-1}\sum_{i=1}^{k-1} P_{q_i}(v)$$

Recall that we only want to consider the previous queries in this manner, while treating the current query differently. Therefore we only sum through the k-1 iteration, as the current query is not treated as history. We must also check that this model is a proper language model (i.e. does the sum over all v equal 1); we desire that:

(2) $$\sum_v P_{qh(k)}(v) = \frac{1}{k-1}\sum_{i=1}^{k-1}\left(\sum_v P_{q_i}(v)\right) = 1$$

Does equation **(2)** hold true? If the query language models, each P_{q_i}, are correct, then $\sum_v P_{q_i}(v) = 1$. Then $\frac{1}{k-1}\sum_{i=1}^{k-1} 1 = \frac{k-1}{k-1} = 1$. Thus, the sum over all *v* does equal 1, and we can consider this model a proper language model. We will also use this equation for the click-through data.

(3) $\sum_v P_{ct(k)}(v) = \frac{1}{k-1}\sum_{i=1}^{k-1}\left(\sum_v P_{ct_i}(v)\right) = 1$

Note, incidentally the correspondence of this model to a VSM-like approach:

$$d \leftrightarrow \left(P_d\left(v^{(1)}\right), \ldots, P_d\left(v^{(m)}\right)\right)^{\mathrm{T}}$$

In the VSM, new "document" vectors can be created by summing or averaging existing document vectors, whereas in the LM approach, new "document" LM's can be created by averaging existing document LM's.

c. 2nd idea: Use Weighted Average

We observe that we may want to weight the contributions from more recent queries (or clickthrough data) higher than the contributions from the earlier feedback history. This thought brings us to our 2nd idea.

Consider how we might use a weighted average of the LM's from the current data and the history. We can simply weight the current query's contribution differently from the history data's contribution:

(4) $P_k^*(v) = \alpha \cdot P_{q_k}(v) + (1-\alpha) \cdot (\text{history model})$

We also may want to weight the query history and clickthrough data differently, so the history model is:

(5) $(\text{history model}) = \beta \cdot P_{qh(k)}(v) + (1-\beta) \cdot P_{ct(k)}(v)$

(6) $P_k^*(v) = \alpha \cdot P_{q_k}(v) + (1-\alpha) \cdot \left(\beta \cdot P_{qh(k)}(v) + (1-\beta) \cdot P_{ct(k)}(v)\right) \qquad \alpha, \beta \in [0,1]$

We observe that **(6)** looks like the Rocchio derivation (with positive feedback only) for the VSM. Therefore, how is this model different from Rocchio's? The difference is in how they score documents' relevance to the query. Recall the basic VSM scoring function (ignoring idf and normalization):

(7) $\sum_j tf_j(q) \cdot tf_j(d) = \sum_j q_j \cdot d_j$

In contrast, the LM scoring function is the Kullback-Leihler divergence between P_k^* and P_d, $D(P_k^* \parallel P_d)$. The divergence is defined by:

(8) $D(P_k^* \parallel P_d) = \sum_v P_k^*(v) \cdot \log \frac{P_k^*(v)}{P_d(v)}$

Note: $\log \frac{P_k^*(v)}{P_d(v)}$ is a log likelihood ratio whose absolute value measures the difference between $P_k^*(v)$ and $P_d(v)$ for specific *v*. Thus **(8)** is an expected log likelihood ratio with respect to the P_k^* distribution. **(8)** can be rewritten as the following:

(9) $D(P_k^* \parallel P_d) = \sum_v P_k^*(v) \cdot \log P_k^*(v) + \sum_v P_k^*(v) \cdot \log \frac{1}{P_d(v)} \xrightarrow{ranking} \sum_v P_k^*(v) \cdot \log \frac{1}{P_d(v)}$

In **(9)**, $\sum_v P_k^*(v) \cdot \log P_k^*(v)$ is the negative of the entropy of P_k^*, the inherent uncertainty of knowing whether the r.v. $V = v$, assuming $V \sim P_k^*$. The last term in **(9)** represents an expected "loss", wrt P_k^*, of using P_d as the basis for describing V. If $P_d = P_k^*$, then this latter term is the entropy of P_k^* and $D(P_k^* \parallel P_d) = 0$ (which is a property you would want of a "distance" function). Perhaps the most important point is that the KL divergence is commonly used to measure the "deviation" between two language models or, in general, two distributions. But also, $-\sum_v P_k^*(v) \cdot \log P_d(v) = \sum_v P_k^*(v) \cdot (-\log P_d(v))$ is reminiscent of the VSM scoring function.

We are left with the question of how to find good values for α and β, which brings us to our 3rd idea.

d. 3rd idea: "Principled" Values for α

Recall our 2nd idea for using implicit feedback. We considered calculating $P_k^*(v)$ using the following formula:

(10) $P_k^*(v) = \alpha \cdot P_{q_k}(v) + (1 - \alpha) \cdot (\text{history model}) \qquad \alpha \in [0,1]$

Previously, we did not specify any procedure for calculating the value for α. One observation is that if our query is relatively long, we may want α to be relatively large, since a longer query could provide a more accurate description of the user's information need than a shorter query.

Here is one formula that turns out to incorporate the length of the query into the value of α:

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

The motivation is to treat the query as observed data and the history as providing (Dirichlet) prior information. We consider $P_{q_k}\left(v^{(j)}\right) = \frac{tf_j(q_k)}{|q|}$, since we are turning term frequencies into a probability.

(11) is a similar formulation to **(10)**, except this time we have represented α differently. When we isolate the first term from **(11)**, we observe that it can be rewritten as:

$$\frac{|q|}{|q| + \mu + \lambda} \cdot P_{q_k}(v) = \alpha \cdot P_{q_k}(v) \qquad \alpha = \frac{|q|}{|q| + \mu + \lambda} \qquad 1 - \alpha = \frac{\mu + \lambda}{|q| + \mu + \lambda}$$

However, we once again have several parameters, namely μ and λ, that we need to estimate. Perhaps we can reduce the amount of parameters we need by using a slightly different concept of "aging" the contributions of our history data, which brings us to our 4th idea.

e. 4th idea: Aging

Assume that in iteration *k*, we have observed query q_k, and we know that $P_{k-1}^*(v)$ is our previously computed prior probability of seeing term *v* based on our previous query and clickthrough history. (We have ignored a discussion made in the original paper regarding whether ct_k is incorporated or not).

Then we can calculate $P_k^*(v)$ using the following, treating P_{k-1}^* as a prior and the current query as observed data:

(12) $$P_k^*(v) = \frac{tf_j(q_k) + \mu \cdot P_{k-1}^*(v)}{|q| + \mu}$$

At each iteration, we combine our information from the previous iteration with that from the current iteration. The information from previous iterations, $P_{k-1}^*(v)$, is weighted by μ. Note that the contribution of early history is damped as we engage in more iterations. We may consider this a form of incremental update.

f. Results

Because this model incorporates implicit feedback, we needed iterations from the user's reformulations of the query. Therefore, the queries chosen had to be relatively hard.

The experimental setup used by Shen, Tan, and Zhai involved 30 hard TREC queries and 3 people generating implicit feedback data. Each person had to try at least 4 queries for each information need, each of the TREC queries, which means they needed to reformulate each query at least 3 times.

Findings:

Notation:
P_{q_1} is the model generated for query formulation 1 for a particular information need.
$P^*_{q_2}$ is the model generated for query formulation 2 for a particular information need, incorporating implicit feedback from the queries and clickthrough data obtained for this information need prior to the user issuing query formulation 2.

Query Reformulations		Queries with Implicit Feedback Adjustment		
P_{q_1}				
P_{q_2}	$\rightarrow$	$P^*_{q_2}$	$-$	not a great improvement (not enough feedback info for good inference)
P_{q_3}	$\rightarrow$	$P^*_{q_3}$	$+$	relatively good improvement
P_{q_4}	$\rightarrow$	$P^*_{q_4}$	$+$	

A few points of interest from the evaluation:

- Clickthrough data was more important than the query history
- Aging worked well for the query history, but badly for the clickthrough data
- Only about 30% of documents clicked were actually relevant, which implies that the summaries were more useful (for feedback purposes) than the actual corresponding documents.

Related Questions

1.

Say we have a given information need. Our vocabulary will be restricted to the following words:

author, book, cat, hat

Our information need is:
Who is the author of "Cat In The Hat"?

Here are the queries, issued in order:

q_1 = "book cat in the hat"
q_2 = "author cat in the hat"

Consider a single document that we want to score:

d = "the author of the book Cat In The Hat is Dr. Seuss"

Calculate the score of the document using formula **(9)** (under ranking). Calculate $P_k^*(v)$ using formula **(11)**. Let μ = .3 and λ = 0. For calculating $P_d(v)$, assume that we don't use smoothing.

Answer:

$$P_{q_1}^*(author) = \frac{0}{3} = 0 \qquad P_{q_1}^*(book) = P_{q_1}^*(cat) = P_{q_1}^*(hat) = \frac{1}{3}$$

$$P_{q_2}^*(author) = \frac{1 + .3 \cdot 0}{3 + .3} = 0.303$$

$$P_{q_2}^*(cat) = P_{q_2}^*(hat) = \frac{1 + .3 \cdot \frac{1}{3}}{3 + .3} = 0.333$$

$$P_{q_2}^*(book) = \frac{0 + .3 \cdot \frac{1}{3}}{3 + .3} = 0.030$$

$$P_d(v) = 0.250,\ \ln[1/P_d(v)] = 1.39$$

Therefore,

$$\sum_v P_{q_1}^*(v) \cdot \log_e \frac{1}{P_d(v)} = 1.39$$

$$\sum_v P_{q_2}^*(v) \cdot \log_e \frac{1}{P_d(v)} = 1.39$$

2.

Why are the scores the same from your answer to question 1?

Answer:

They are the same because the document induces a uniform distribution (maximum entropy) over the vocabulary. Thus, $P_d(v)$ is constant (since we didn't use smoothing), and each $P_k^*(v)$ is given the same amount of weight for all *v*.

$$\sum_v P_{q_i}^*(v) \cdot \log \frac{1}{P_d(v)} = \sum_v P_{q_i}^*(v) \cdot const = const \cdot \sum_v P_{q_i}^*(v) = const \cdot 1 = const$$

3.

At the end of examining our 1st idea, we mentioned an intuition that we may want to weight contributions from more recent queries higher than those from earlier in the history. Give an example of a problem in the way queries can be generated by users (i.e. reformulated) that could cause this intuition to adversely affect our performance.

Answer:

Consider a case of a short session with a search engine for a single information need where the task is very difficult. The user is continually reformulating queries in order to meet their need. Then our intuition of weighting contributions from more recent queries higher than those from earlier queries relies on the assumption that queries generated later in time are "better" than queries generated earlier. However, it may be the case that the user knows little about the system, causing the queries they generate by reformulation to fetch fewer relevant documents over time. We already observed that some users know little about how to pick good terms (terms helpful to system performance).

4.

In our results discussion, we mentioned that clickthrough data is more important, but it does not actually become more useful when aging is applied. While we only considered summaries (or the corresponding documents) clicked on to compose our clickthrough history in our discussion, we can also imagine using all data from a user's email and desktop action history as additional clickthrough data (see Teevan, Dumais, Horvitz '05).

One problem with many queries is that they don't provide the system with enough information to easily disambiguate words that have multiple meanings and associations. How might extensive clickthrough data help our system even if it isn't aged?

Answer

Consider the case of a statistician searching for the statistician Michael Jordan's homepage with the query "Michael Jordan". The statistician will be presented with a lot of links to sites related to Michael Jordan the basketball player. However, if our system incorporated the emails from statistical conferences and the statistical papers that our user has on their desktop as clickthrough data, then it could instead return links to sites related to Michael Jordan the statistician. In short, it gives us a crude form of user profile (assuming the statistician doesn't receive too much spam…).