CS 630 Lecture 8: 02/21/2006
Lecturer: Lillian Lee Scribes: Peter Babinski, David Lin

Basic Language Modeling Approach

I. Special Case of LM-based Approach

a. Recap of Formulas and Terms

Recall our special case of a language modeling-based approach. We decided to use multinomial "topic" models with corpus-dependent Dirichlet priors. We wanted to score a document *d* by using the probability of the query based on a model based on the document, $P_d(q)$. Using a multinomial model gives us the following equation for $P_d(q)$, with respect to same length term sequences.

(1) $P_d(q) = K(q) \times \prod_j \left[\theta_j(d)\right]^{tf_j(q)}$

$K(q)$ represents the number of ways we can rearrange the terms of the query. Recall the following equation for $\theta_j(d)$:

(2) $\vartheta_j(d) = \left[\dfrac{tf_j(d) + \mu \times tf_j(C)/|C|}{|d| + \mu}\right]$

Recall that we incorporated this smoothing term into $\theta_j(d)$:

(3) $\mu \times tf_j(C)/|C|$

We added **(3)** to the equation for $\theta_j(d)$ in order to avoid situations where we would have zero counts for a particular term $v^{(j)}$ in our vector representation of the document. Having zero counts for any component of the product sum in the equation for $P_d(q)$ for which any of the corresponding counts in the query are non-zero would naturally cause our probability to become zero, so we added counts to each term based upon its frequency of occurrence in the corpus.

We then remove the document independent term, $K(q)$, because it does not affect our ranking of documents, in order to obtain the following scoring equation for $P_d(q)$:

(4) $$P_d(q) \xrightarrow{\text{rank}} \cdot \prod_j \left[\frac{tf_j(d) + \mu \times tf_j(C)/|C|}{|d| + \mu} \right]^{tf_j(q)}$$

Notice that many familiar terms appear in **(4)**. $tf_j(d)$ is our term frequency component. $|d|$ is our length normalization adjustment. $tf_j(C)/|C|$ looks like some form of a reciprocal IDF term.

This new equation seems a bit problematic. The smoothing function in the equation adds counts to a term proportional to its frequency of occurrence, the exact opposite of the way in which the IDF measure weights terms. A term such as "the" would gain many counts from this smoothing function. We would have preferred to see a term resembling the IDF quantity appear.

b. Fixing θ_j?

There are a few ways we could try to fix θ_j, shown in **(2)**:

- We could remove the smoothing term (set μ = 0), but this would result in our estimation treating a document that is missing one query term identically to how it treats documents missing more than one (or even all) query terms.
- We could also change the smoothing term to incorporate a uniform prior. The resulting LM estimation would be different, and it doesn't take advantage of all of the information we have available from the corpus. In addition, we have little justification for this choice.
- We could back off from the generative approach entirely.

However, before we decide to take any action, shouldn't we see if there is any way that we can manipulate **(4)** to give us the IDF term we want? In other words, is the IDF term already in **(4)**?

Our approach, following Zhai and Lafferty (2001), will be to try to drive $tf_j(C)$ into the denominator. First we define the following:

(5) $$norm(d) = norm(d, \mu, q) = \left(|d| + \mu\right)^{|q|}$$

Using this quantity, we can rewrite **(4)**:

(6) $$P_d(q) = \frac{1}{norm(d)} \times \prod_j \left[tf_j(d) + \mu \times tf_j(C)/|C| \right]^{tf_j(q)}$$

Recall our RSJ derivation. We can use some simple transformations to try to separate missing terms. We begin by considering splitting the product sum according to the indices *j*. One product sum will only include indices *j* such that $tf_j(d) > 0$. The other product sum will only include indices *j* such that $tf_j(d) = 0$. Thus we obtain the following equation from **(6)**:

(7) $$\frac{1}{norm(d)} \times \prod_{j:tf_j(d)>0} \left[tf_j(d) + \mu \times tf_j(C)/|C| \right]^{tf_j(q)} \times \prod_{j:tf_j(d)=0} \left[\mu \times tf_j(C)/|C| \right]^{tf_j(q)}$$

Notice that the RHS of **(7)** is independent of *d* except for in the index. We can eliminate this RHS by multiplying by this term:

(8) $$\prod_{j:tf_j(d)>0} \left[\mu \times tf_j(C)/|C| \right]^{tf_j(q)}$$

Because we now have a product sum over all indices *j* of this term that is independent of *d*, we can eliminate it under ranking.

(9) $$\prod_{j:tf_j(d)=0} \left[\mu \times tf_j(C)/|C| \right]^{tf_j(q)} \times \prod_{j:tf_j(d)>0} \left[\mu \times tf_j(C)/|C| \right]^{tf_j(q)} = \prod_j \left[\mu \times tf_j(C)/|C| \right]^{tf_j(q)}$$

The resulting term in **(9)** will not affect our ranking, so we can eliminate it. However, we also need to multiply by the reciprocal of **(8)** in order to maintain equality.

(10) $$\frac{1}{norm(d)} \times \prod_{j:tf_j(d)>0} \left[tf_j(d) + \mu \times tf_j(C)/|C| \right]^{tf_j(q)} \times \prod_{j:tf_j(d)=0} \left[\mu \times tf_j(C)/|C| \right]^{tf_j(q)} \times$$
$$\prod_{j:tf_j(d)>0} \left[\mu \times tf_j(C)/|C| \right]^{tf_j(q)} \times \prod_{j:tf_j(d)>0} \left[|C|/\mu \times tf_j(C) \right]^{tf_j(q)}$$

After multiplying through and noting our equivalence in **(9)** does not affect our ranking, we obtain the following:

(11) $$P_d(q) \xrightarrow{ranking} \frac{1}{norm(d)} \times \prod_{j:tf_j(d)>0} \left[\frac{tf_j(d) \times |C|}{\mu \times tf_j(C)} + 1 \right]^{tf_j(q)}$$

We still have our term frequency, $tf_j(d)$, and our length normalization, $|d|$. In addition, we now have our IDF term, $|C|/tf_j(C)$.

c. About that Multinomial Model…

Let us first review some of the aspects and motivations of our language model.

- We wanted to model a probabilistic text-generation process for the query based somehow on the document.
- We wanted a method for assigning probabilities to text.

While there are a variety of sophisticated models we could have chosen, we decided upon a multinomial one.

Aside: Our multinomial model is not necessary the same as a unigram model, as considering the two equal creates some "checksum" issues. For example:

Assume parameters $\theta_1,\ldots,\theta_m$ s.t. $\sum_j \theta_j = 1$ and for all j, $\theta_j > 0$.

Thus "$P_U(q)$" = $\prod_j [\theta_j]^{tf_j(q)}$, where "$P_U(q)$" is our probability of the query given some unigram model, assuming independence of occurrences for each term j. For the probability of any term $v^{(j)}$, we know the following:

"$P_U(v^{(j)})$" = θ_j

Now these probabilities must sum to 1.

$$(*) \quad \sum_j "P_U(v^{(j)})" = \sum_j \theta_j = 1$$

However, we now note that there is no probability left for multiple terms occurring.

$$P_U(v^{(1)}v^{(2)}) = \begin{cases} \theta_1 \times \theta_2 & \text{expected result} \\ 0 & \text{because of } (*) \end{cases}$$

We expected $P(v^{(1)}v^{(2)}) = \theta_1 \times \theta_2$ by our above definition of $P_U(q)$. However, we found that $P_U(v^{(1)}v^{(2)}) = 0$ due to the fact that $\sum_j P_U(v^{(j)}) + P_U(v^{(1)}v^{(2)}) \le 1$ by the rules of probability.

How is our multinomial model different? $P_d(q) = K(q) \times P_U(q)$
The quantity $K(q)$ restricts our query to a given length. However, the two quantities are equivalent under ranking, as the $K(q)$ term is document independent.

Let us see how we could fix the unigram model. We will consider a 2-state HMM (Hidden Markov Model).

We incorporate s, the probability of stopping generation of the text. Therefore we observe probabilities are calculated as the following:

$$P_H(v^{(1)}) = (1-s)\theta_1 s$$
$$P_H(v^{(1)}v^{(2)}) = (1-s)\theta_1(1-s)\theta_2 s$$
$$P_H(v^{(i_1)}v^{(i_2)}\ldots v^{(i_k)}) = (1-s)^k s\prod_j^k \theta_{i_j}$$

Claim: Under this model, we obtain a proper distribution.

II. A New Interpretation of our LM Approach

a. A New Scenario

Recall we originally wanted our documents' relevance to the query using the following:

(12) $P(R = y \mid D = d, Q = q)$

Our special case derivation resulted in a VSM-like model. During our derivation in previous lectures of the "query-likelihood" scoring function we used, we at some point finessed the concept of selection vs. generation. The reasoning we used here wasn't as justifiable as we would have liked.

Let us try to backtrack a bit from our previous thinking in order to find a better explanation for how we got to the point we derived earlier in our special case. In a new scenario, we want to try a new interpretation of what $D = d$ means.

Assume:

- The corpus contents are described by a set of "topic" LM's.
- Documents are generated by a choice of topic (T_D), which then generates d (D)
- There is also a set of "info need" LM's (these could be user-specific perhaps?)
- The query is created by a choice of topic (T_Q), which then creates q (Q)
- Therefore the concept of relevance, $R = y$, requires that $T_D = t = T_Q$.

Some immediate questions come to mind. What happens if we have documents that are about multiple topics? But note that the notion of "topic" as used here is also somewhat flexible, and so we can (partially) accommodate such a situation.

The remaining portion of our LM discussion will be continued next lecture.

III. Related Questions

Let us examine how the ranking function we derived from our LM model behaves with some sample data. Recall the equation we use for ranking is found in **(11)**.

Query:
"tips on bass fishing" length = 4

Documents:
d(1) = "fishing bass for fun" length = 4
d(2) = "tips on fishing" length = 3
d(3) = "fishing for tips as a waiter" length = 6

We will consider the set of documents to be our corpus C for the smoothing term.
Let $\mu = .5$.

1. Calculate $P_d(q)$ using **(11)**.

Answer:

First the norms for each document:

$norm(d(1)) = (4 + .5)^4 = 410.1$
$norm(d(2)) = (3 + .5)^4 = 150.1$
$norm(d(3)) = (6 + .5)^4 = 1785.1$

Terms ($j : tf_j(d) > 0$):

fishing, bass, for, fun, tips, on, as, a, waiter

Raw term counts:

fishing = 3, bass = 1, for = 2, fun = 1, tips = 2, on = 1, as = 1, a = 1, waiter = 1

|C| = 4 + 3 + 6 = 13

Now the product sums:

$$fishing\ term \times bass\ term \times tips\ term \times on\ term$$

$$P_{d(1)}(q) = \left[\frac{1\times 13}{.5\times 3}+1\right]^1 \times \left[\frac{1\times 13}{.5\times 1}+1\right]^1 \times \frac{1}{410.1} = .636$$

$$P_{d(2)}(q) = \left[\frac{1\times 13}{.5\times 3}+1\right]^1 \times \left[\frac{1\times 13}{.5\times 2}+1\right]^1 \times \left[\frac{1\times 13}{.5\times 1}+1\right]^1 \times \frac{1}{150.1} = 24.344$$

$$P_{d(3)}(q) = \left[\frac{1\times 13}{.5\times 3}+1\right]^1 \times \left[\frac{1\times 13}{.5\times 2}+1\right]^1 \times \frac{1}{1785.1} = .076$$

2. Recall that for our special case, we decided to use Dirichlet smoothing to aid is in estimating the quantity $P_d(q)$. However, Dirichlet smoothing is not the only possibility. What if we were to try another smoothing method? Say we decided to use linear interpolation to smooth each $\theta_j(d)$.

How would the formula in **(2)** change? Rewrite $P_d(q)$ using your new formula for $\theta_j(d)$.

Answer:

$$\theta_j(d) = \lambda \frac{tf_j(d)}{|d|} + (1-\lambda)\frac{tf_j(C)}{|C|}$$

$$P_d(q) = \prod_j \left[\lambda \frac{tf_j(d)}{|d|} + (1-\lambda)\frac{tf_j(C)}{|C|}\right]^{tf_j(q)}$$

3. Think about how you would calculate $P_d(q)$ using the new smoothing method from question 2. We don't seem to have a term that acts like an IDF in our current formula for $P_d(q)$. Show that the IDF term is in fact present when we use $P_d(q)$ to rank documents.

Answer:

When we examine $P_d(q)$ from question 2 under ranking, we observe the following:

$$\textbf{(13)}\ P_d(q) = \prod_{j:tf_j(d)>0}\left[\lambda \frac{tf_j(d)}{|d|} + (1-\lambda)\frac{tf_j(C)}{|C|}\right]^{tf_j(q)} \times \prod_{j:tf_j(d)=0}\left[\lambda \frac{tf_j(d)}{|d|} + (1-\lambda)\frac{tf_j(C)}{|C|}\right]^{tf_j(q)}$$

Since each $tf_j(d)$ count in the second term in equation **(13)** is equal to zero, we observe that that product sum simplifies as follows:

(14) $$\prod_{j:tf_j(d)=0}\left[\lambda\frac{tf_j(d)}{|d|}+(1-\lambda)\frac{tf_j(C)}{|C|}\right]^{tf_j(q)}=\prod_{j:tf_j(d)=0}\left[(1-\lambda)\frac{tf_j(C)}{|C|}\right]^{tf_j(q)}$$

Now the second term in equation **(13)**, shown in **(14)**, is almost document independent except for its index set. We can make **(14)** document independent by multiplying **(13)** by the same term over the remaining index set, $j:tf_j(d)>0$.

(15) $$\prod_{j:tf_j(d)>0}\left[(1-\lambda)\frac{tf_j(C)}{|C|}\right]^{tf_j(q)}$$

However, in order to maintain equality, we must also multiply **(13)** by the reciprocal of **(15)**. Here is our equation for $P_d(q)$ after the aforementioned transformations.

(16)

$$P_d(q)=\prod_{j:tf_j(d)>0}\left[\lambda\frac{tf_j(d)}{|d|}+(1-\lambda)\frac{tf_j(C)}{|C|}\right]^{tf_j(q)}\times\prod_{j:tf_j(d)=0}\left[(1-\lambda)\frac{tf_j(C)}{|C|}\right]^{tf_j(q)}\times\prod_{j:tf_j(d)>0}\left[(1-\lambda)\frac{tf_j(C)}{|C|}\right]^{tf_j(q)}\times$$

$$\prod_{j:tf_j(d)>0}\left[\frac{1}{\left[(1-\lambda)\frac{tf_j(C)}{|C|}\right]}\right]^{tf_j(q)}$$

Simplifying, the middle two terms of the product are document independent, so we can remove them under ranking.

(17) $$P_d(q)\xrightarrow{ranking}\prod_{j:tf_j(d)>0}\left[\lambda\frac{tf_j(d)}{|d|}+(1-\lambda)\frac{tf_j(C)}{|C|}\right]^{tf_j(q)}\times\prod_{j:tf_j(d)>0}\left[\frac{1}{\left[(1-\lambda)\frac{tf_j(C)}{|C|}\right]}\right]^{tf_j(q)}=$$

$$\prod_{j:tf_j(d)>0}\left[\lambda\frac{tf_j(d)}{|d|}\times\frac{|C|}{(1-\lambda)\cdot tf_j(C)}+1\right]^{tf_j(q)}$$