CS 6740: Advanced Language Technologies February 25, 2010

Lecture 9: Language Models: More on Query Likelihood

Lecturer: Lillian Lee *Scribes: Navin Sivakumar, Lakshmi Ganesh, Taiyang Chen*

Abstract

The Language Modeling approach to document retrieval scores the relevance of a document d with respect to a query q as the likelihood that a language model induced from d would generate q. This *query-generative* model is counter-intuitive; after all, we are looking for a retrieval system that takes a query as input and generates relevant *documents*; yet, here we appear to be trying to generate the query itself. In this lecture we examine the query-generative perspective of the language modeling approach in more detail and attempt to justify it.

1 Review of the Language Modeling Approach

First, we briefly review the language-modeling approach to scoring documents [PC98]. Recall that if $\vec{x}$ denotes an m-dimensional vector, then $x[j]$ denotes the jth entry and we write $x[\cdot]$ for $\sum_{j=1}^{m} x[j]$. Given a document d, we consider the language model (a probability distribution over strings of terms) induced by the document. The score of a document d with respect to a query q is given by the probability assigned to q by the language model induced by d. More formally, a document d induces a vector $\vec{\theta}_d$ of parameters. We then score d with respect to a query q by $P_{\vec{\theta}_d}(\vec{q})$, where $P_{\vec{\theta}_d}$ denotes the probability distribution specified by the parameters $\vec{\theta}_d$, and $\vec{q}$ is the vector of term counts[1] in the query q. Typically, we consider multinomial distributions $P_{\vec{\theta}}$, which are parametrized by a length parameter L, specifying the number of trials (in our case, the length of the string being generated), and the probabilities $\theta[j]$ of term v_j occurring in each trial. This gives us a scoring function of the form

$$P_{\vec{\theta}}(\vec{q}) = k \prod_j \theta[j]^{q[j]} \tag{1}$$

where k is a constant (independent of the $\theta[j]$ parameters) giving the number of possible rearrangements of the terms in q. Since k is independent of the parameters that arise from the document d, the following is equivalent under rank:

$$P_{\vec{\theta}}(\vec{q}) \stackrel{rank}{=} \prod_j \theta[j]^{q[j]} \tag{2}$$

[1]From time to time we are somewhat loose in interpreting a language model as either a distribution on strings of terms or a distribution on term-count vectors.

From a document d we induce the parameter $\theta_d[j]$ through the rate of occurrence of term v_j in d:

$$\theta_d[j] = \frac{f_d[j]}{f_d[\cdot]} \tag{3}$$

With Dirichlet smoothing [MP95], we can induce the TF, IDF and length normalization terms.

2 Why "Query Likelihood"?

Let us discuss why it is appropriate to consider the query as the object being generated by our model. As a point of comparison, we consider the alternative approach of inducing a language model from a query and considering the document as the object being generated.[2] This approach to scoring documents has a plausible interpretation as giving documents a high score if they are likely to be generated by the language model given by a query.

However, there are two immediate weaknesses to this approach:

- One practical concern is that queries are typically much shorter than documents; therefore, queries provide much less data than documents for us to use in constructing a language model. Despite this, we can generate a term probability $\theta[j]$ for each term v_j based on the frequency of v_j in q (possibly with some smoothing).
- Another problem is that, for a given query, the query length is fixed, whereas document lengths vary over the corpus. This means that a query-generative model is easier to work with than a document-generative one. For example, if the query q consisted of the single term v_1, then the (unsmoothed) probability distribution over terms induced from this query is given by: $\vec{\theta_q}[1] = 1$. Now we run up against an inconsistency: $P_{\vec{\theta_q}}(v_1) = P_{\vec{\theta_q}}(v_1 v_1) = \ldots = P_{\vec{\theta_q}}(v_{1_1} v_{1_2} \ldots v_{1_{|d|}}) = 1$. Our probabilities sum to more than one! What this example reveals is that in our query-generative model there was an implicit length parameter, which becomes explicit (and problematic) in the document-generative world.

We can attempt to resolve the latter issue by using an alternative, more complicated language model. Consider the finite Markov chain shown in Fig 1. We can interpret the document generation process as follows:

1. The document ends with probability p_{stop}; otherwise proceed to step 2.
2. Add a term to the document with probabilities given by $\vec{\theta_q}$; return to step 1.

[2] Recall from the last lecture the post-hoc justification in [LZ02] for query likelihood as the outcome of performing a different Bayes flip of r and q in the derivation of the probabilistic model of [RJ76]; the "document likelihood" approach we are presently considering can be thought of as the result of instead performing a Bayes flip of d and r.

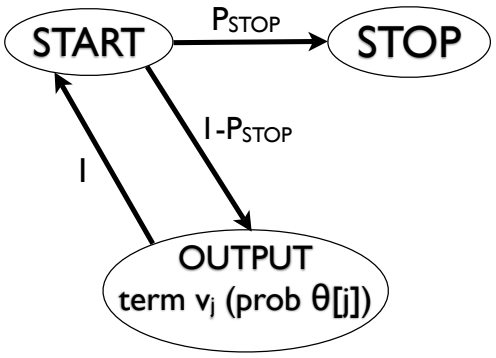

Figure 1: Language Model Based On a Finite Markov Model

We can then explicitly write the scoring function for a document d:

$$\begin{aligned} P_{\vec{\theta}_q, p_{\text{stop}}}(d) &= \left(\prod_j \theta_q[j]^{f_d[j]} (1 - p_{\text{stop}})^{f_d[j]} \right) p_{\text{stop}} \\ &= p_{\text{stop}} (1 - p_{\text{stop}})^{f_d[.]} \prod_j \theta_q[j]^{f_d[j]} \end{aligned} \tag{4}$$

However, this scoring function heavily penalizes long documents, which have larger $f_d[.]$ values. If we want to avoid penalizing long documents, we must take a different approach.

3 Relevance Models

One of the criticisms of the language modeling approach is its counter-intuitive query-generative perspective. We have discussed the issues with using a document-generative model instead. An alternative is to drop generative models altogether and return to an approach where *relevance*, not the query, takes the forefront (as in the Robertson-Spärck Jones probabilistic model [RJ76]). On that note, let us consider the model of [LC01], a relevance-based and empirically effective model. We will see that even when we begin from a relevance-centric perspective, we are naturally led back to the query-generative world; thus, this exercise provides further evidence for the legitimacy of the query-generative approach.

Recall that in the derivation of the probabilistic scoring function for [RJ76] with binary attributes, we had the following weight for terms v_j such that both $f_d[j] > 0$ and $q[j] > 0$ (1 in the case of binary attributes):

$$\frac{P(F[j] = 1 \mid r, \vec{q})}{P(F[j] = 1)}.$$

Analogously, [LC01] uses a term weight of the form

$$\frac{P(v_j \mid r, \vec{q})}{P(v_j)},$$

where $P(v_j \mid r, \vec{q})$ should be interpreted as the rate of occurrence of term v_j in relevant documents.

How should we estimate $P(v_j \mid r, \vec{q})$? Consider a fixed query q. If we knew the set of relevant documents **Rel**, we could approximate the probability with the average rate of occurrence of term v_j in all relevant documents as follows:

$$\begin{aligned} P(v_j \mid r, \vec{q}) &\approx \sum_{d \in \mathbf{Rel}} P(d \mid \mathbf{Rel}) \frac{\#\,(v_j \text{ in } d)}{\text{length of } d} \\ &= \frac{1}{|\mathbf{Rel}|} \sum_{d \in \mathbf{Rel}} \frac{\#\,(v_j \text{ in } d)}{\text{length of } d} \\ &= \frac{1}{|\mathbf{Rel}|} \sum_{d \in \mathbf{Rel}} P_{\vec{\theta}_d}(v_j) \end{aligned} \tag{5}$$

where we have made the observation that language models induced by documents occur naturally in this approximation.

Of course, we typically do not know the set of relevant documents **Rel**. In this case, we make the assumption that there is a common underlying language model that generates the query and relevant documents. Then we would like to weight term v_j according to the probability of v_j in this underlying model. Since we know nothing about the set of relevant documents, the only evidence we can use to construct this underlying model is the fact that it can generate the query q. We can capture this intuition as follows:

$$\begin{aligned} P(v_j \mid \mathbf{Rel}) &\approx P(v_j \text{ is next word} \mid \text{previous words were } q) \\ &= \frac{P(qv_j)}{P(q)} \end{aligned} \tag{6}$$

where $P(q) = \sum_{j'} P(qv_{j'})$. Note that all the strings $qv_{j'}$ have the same length, so we can use multinomial distributions to approximate the probabilities without worrying about the length parameter. Two suggested approaches for computing $P(qv_j)$ are given below (let L denote the length of the query):

1. Average over all documents the rate of occurrence of the string qv_j:

$$P(qv_j) = \sum_d P(d) P_{\vec{\theta}_d}(qv_j) = \sum_d P(d) P_{\vec{\theta}_d}(v_j) \prod_{k=1}^{L} P_{\vec{\theta}_d}(q_k) \tag{7}$$

2. For each query term q_k, average over documents (according to probabilities given by v_j) the rate of occurrence of q_k. We then compute the probability of qv_j by multiplying the prior probability of v_j by the averaged rates of occurrence of each query term q_k:

$$P(qv_j) = P(v_j) \prod_{k=1}^{L} \sum_d P(d \mid v_j) P_{\vec{\theta}_d}(q_k) \tag{8}$$

In either case, we return to the notion of query-generation by language models induced by documents.

4 Exercises

1. Recall that the language-modeling approach [PC98] scores a document d with respect to a query q by the probability that the language model induced by d generates the query q, sometimes written as $P(q \mid d)$. The fact that we use a probability measure over queries[3] to score documents forces the scoring function to satisfy certain properties. For example, consider two documents d_1 and d_2; if there is a query q such that d_1 scores higher than d_2, then there must be another query q' such that d_1 scores lower than d_2, i.e.

$$P(q \mid d_1) > P(q \mid d_2) \Rightarrow \exists q'.P(q' \mid d_1) < P(q' \mid d_2).$$

   (a) Prove that the property above holds.

   (b) Discuss whether this is a desirable property for a scoring function.

2. (a) Recall the query-likelihood scoring function for a document d with respect to query q:

$$\prod_j \theta_d[j]^{q[j]},$$

   where $\theta_d[j]$ depends on the frequency of term v_j in d. Note that d is rewarded only for containing terms that occur in q; if a term does not occur in q, then $q[j] = 0$, so $\theta_d[j]^{q[j]}$ contributes nothing to the product. Explain why the relevance model scoring function does not have this property; that is, if we use a scoring function based on weighting term v_j by $P(v_j \mid q)$, then it is conceivable for a document's score to be increased by the occurrence of terms that are not in q. Can you give an interpretation for why it might be desirable to allow terms that do not occur in the query to increase a document's score?

   (b) One approach to resolving this issue while still using a language-modeling scoring function is *query expansion*: the scoring system adds terms to the query, thereby allowing terms that were not in the original query to contribute to the document score [XC00]. What might be a plausible approach for performing query expansion?

3. Recall the term weight function of the [LC01] relevance model. Through the derivation in (5) and the elimination of **Rel** in (6), we have reduced the equation to a final term $P(qv_j)$. Two ways (both query-generative) – (7) and (8) – are proposed to evaluate this quantity and thus the scoring function. In this exercise we contrast these two methods.
   Suppose we have the following vocabulary:

v_1	v_2	v_3	v_4
a	big	machine	super

[3] In the language modeling approach as described in the notes, there is a slight complication involving the length parameter of the multinomial distribution induced by d; for now, assume that a document d induces a probability distribution over all queries (of any length).

Consider the following corpus of documents:

d_1	big super machine
d_2	big big big big machine
d_3	a machine super machine

Assume that we use the simple multinomial language model (2) with probabilities defined by (3).
For the following queries,

q_1	a super machine
q_2	super big

compare the relevance scores derived using (7) and (8). Discuss whether the results correspond to your intuition about which documents are relevant to which queries. Does the example illustrate any differences between the two scoring methods?

5 Solutions

1. (a) We are given that $P(q \mid d_1) > P(q \mid d_2)$. Suppose that for all q' distinct from q, $P(q' \mid d_1) \geq P(q' \mid d_2)$. Then we have

$$\begin{aligned} 1 &= \sum_{q'} P(q' \mid d_1) \\ &= P(q \mid d_1) + \sum_{q' \neq q} P(q' \mid d_1) \\ &> P(q \mid d_2) + \sum_{q' \neq q} P(q' \mid d_2) \\ &= \sum_{q'} P(q' \mid d_2) = 1 \end{aligned}$$

   which is a contradiction. Hence there exists q' such that

$$P(q' \mid d_1) < P(q' \mid d_2).$$

   (b) If one believes that some documents are absolutely more relevant (regardless of the query) than other documents, then this is an undesirable property for a scoring function. Consider the case of an original document d and a modified document d' resulting from a factual correction in d; it seems plausible that one would want d' to score higher than d regardless of the query. A more extreme example is a "spam" document that a reader would not find relevant to any query; this might be considered less relevant with respect to any query than a coherent document.
   In addition, recall our discussion in an earlier lecture on the relationship between length and relevance. We found that at least in one annotated

corpus, there was some evidence that longer documents were more relevant (with respect to a collection of queries) than shorter documents [SBM96]. While this is not an example of the strict dominance of one document compared to another, it does suggest that there may be an issue with the "zero-sum" property of a query-likelihood scoring function (i.e. any increase in a document's score with respect to one query must be offset by a decrease in score with respect to another query), since some documents seem to deserve a greater overall "scoring mass" than others.

2. (a) Consider the weight assigned to term v_j in the relevance model:

$$P(v_j \mid q).$$

If term v_j occurs often in the corpus in association with the query q, then v_j receives a large weight, even if it does not occur in q. In that case, a document's score is increased by occurrences of term v_j, which is not a query term.
If a $P(v_j \mid q)$ is large, then v_j is likely to be a word associated with the topic expressed by the query q. Therefore, it is reasonable to reward documents for containing v_j.

(b) Here are a couple of reasonable approaches to query expansion. The basic principle is to add terms to the query that occur frequently in relevant documents. One idea is to use a baseline retrieval method to retrieve documents based on the query q and add terms to q based on their occurrence in the retrieved documents, since the retrieved documents ought to be relevant to the query. This is the basic idea used in [XC00].[4] Note that this approach is dependent on the baseline retrieval method being reasonably effective at identifying relevant documents.
An alternative approach might be to combine the relevance modeling approach with the query-likelihood score function. Note that [LC01] already provides us with a way to identify terms that are related to the query q; namely, we have the weight function $P(v_j \mid q)$, as well as some empirically good estimates for this quantity. Intuitively, this weight ought to tell us the extent to which the presence of v_j in a document d is evidence that d is relevant to q. It seems reasonable then to instead modify the query q by augmenting $q[j]$ by a quantity that is based on (an estimate of) $P(v_j \mid q)$.

3. Note: all the logarithms in the calculations are base 10. Let us use $P_1()$ to denote the probability defined by method 1 (7) and $P_2()$ to denote the probability defined by method 2 (8). Recall that the scoring function is given by,

[4] In fact, this approach has much earlier origins; [XC00] offers some improvements to the earlier attempts and applies them in the context of a language modeling score function.

$$
\begin{aligned}
score_q(d) &= \sum_{v_j \in d \wedge v_j \in q} \log\left(\frac{P(v_j \mid r, \vec{q})}{P(v_j)}\right) \\
&= \sum_{v_j \in d \wedge v_j \in q} \log\left(\frac{P(qv_j)}{\left(\sum_{j'} P(qv_{j'})\right) P(v_j)}\right) \qquad (9)
\end{aligned}
$$

First we find the rate of occurrence $P(v_j)$ for each term in document by averaging the rate of occurrence in each document:

v_j	$P(v_j)$
v_1	$\frac{1}{3}(0) + \frac{1}{3}(0) + \left(\frac{1}{3} \times \frac{1}{4}\right) = \frac{1}{12}$
v_2	$\left(\frac{1}{3} \times \frac{1}{3}\right) + \left(\frac{1}{3} \times \frac{4}{5}\right) + \frac{1}{3}(0) = \frac{17}{45}$
v_3	$\left(\frac{1}{3} \times \frac{1}{3}\right) + \left(\frac{1}{3} \times \frac{1}{5}\right) + \left(\frac{1}{3} \times \frac{1}{2}\right) = \frac{31}{90}$
v_4	$\left(\frac{1}{3} \times \frac{1}{3}\right) + \frac{1}{3}(0) + \left(\frac{1}{3} \times \frac{1}{4}\right) = \frac{7}{36}$

Let $q' = qv_j$. Using method 1 (7), we get:

$$
\begin{aligned}
P_1(q') &= \sum_d P(d) P_{\vec{\theta}_d}(q') \\
&= \sum_d P(d) \prod_j \left(\frac{f_d[j]}{f_d[\cdot]}\right)^{q'[j]} \\
&= \frac{1}{3} \times \left(\frac{0^{q'[1]}}{3^{q'[2]+q'[3]+q'[4]}} + \frac{4^{q'[2]} \times 0^{q'[1]+q'[4]}}{5^{q'[2]+q'[3]}} + \frac{0^{q'[2]} \times 2^{q'[3]}}{4^{q'[1]+q'[3]+q'[4]}}\right)
\end{aligned}
$$

where we adopt the convention $0^0 = 1$.
Plugging in the query terms from our corpus, we get the following results:

q	$P_1(qv_1)$	$P_1(qv_2)$	$P_1(qv_3)$	$P_1(qv_4)$	$\sum_{j'} P_1(qv'_j)$
q_1	0.002604	0	0.005208	0.002604	0.01042
q_2	0	0.01235	0.01235	0.01235	0.03704

Now compute the $\frac{P_1(qv_j)}{\sum_{j'} P_1(qv_{j'})}$ terms:

q	$\frac{P_1(qv_1)}{\sum_{j'} P_1(qv_{j'})}$	$\frac{P_1(qv_j)}{\sum_{j'} P_1(qv_{j'})}$	$\frac{P_1(qv_j)}{\sum_{j'} P_1(qv_{j'})}$	$\frac{P_1(qv_j)}{\sum_{j'} P_1(qv_{j'})}$
q_1	0.2500	0	0.5000	0.2500
q_2	0	0.3333	0.3333	0.3333

We can calculate the individual log terms in the scoring function:

v_j	$\log\left(\frac{P_1(v_j\|r,\vec{q_1})}{P(v_j)}\right)$	$\log\left(\frac{P_1(v_j\|r,\vec{q_2})}{P(v_j)}\right)$
v_1	$\log\left(\frac{0.2500}{\frac{1}{12}}\right) = 0.4771$	$\log\left(\frac{0}{\frac{1}{12}}\right) = -\infty$
v_2	$\log\left(\frac{0}{\frac{17}{45}}\right) = -\infty$	$\log\left(\frac{0.3333}{\frac{17}{45}}\right) = -0.05436$
v_3	$\log\left(\frac{0.5000}{\frac{31}{90}}\right) = 0.1619$	$\log\left(\frac{0.3333}{\frac{31}{90}}\right) = -0.01424$
v_4	$\log\left(\frac{0.2500}{\frac{7}{36}}\right) = 0.1091$	$\log\left(\frac{0.3333}{\frac{7}{36}}\right) = 0.2341$

And so we can finally compute the document scores by adding the term weights from the table above for each term v_j that occurs in both the document and the query:

q	$score_q(d_1)$	$score_q(d_2)$	$score_q(d_3)$
q_1	$0.1619 + 0.1091 = 0.2710$	0.1619	$0.4771 + 0.1619 + 0.1091 = 0.7481$
q_2	$-0.05436 + 0.2341 = 0.1797$	-0.05436	0.2341

Now let's use method 2, represented by equation (8):

$$\begin{aligned}
P_2(qv_j) &= P(v_j)\prod_{k=1}^{L}\sum_{d} P(d \mid v_j)P_{\vec{\theta}_d}(q_k) \\
&= P(v_j)\prod_{k=1}^{L}\sum_{d} P(d \mid v_j)\prod_{j}\left(\frac{f_d[j]}{f_d[\cdot]}\right)^{q_k[j]} \\
&= P(v_j)\prod_{k=1}^{L}\left(\frac{0^{q_k[1]}P(d_1 \mid v_j)}{3^{q_k[2]+q_k[3]+q_k[4]}} + \frac{4^{q_k[2]}0^{q_k[1]+q_k[4]}P(d_2 \mid v_j)}{5^{q_k[2]+q_k[3]}} + \right. \\
&\quad \left. \frac{2^{q_k[3]}0^{q_k[2]}P(d_3 \mid v_j)}{4^{q_k[1]+q_k[3]+q_k[4]}}\right)
\end{aligned}$$

where $q_k[k] = q[k]$, and $q_k[j] = 0$ if $j \neq k$. Again, we adopt the convention $0^0 = 1$.

Note that

$$P(d \mid v_j) = \begin{cases} \frac{1}{\#\text{ of documents containing } v_j}, & \text{if } d \text{ contains } v_j \\ 0, & \text{otherwise} \end{cases}$$

The values of $P(d \mid v_j)$ are given below:

d	$P(d \mid v_1)$	$P(d \mid v_2)$	$P(d \mid v_3)$	$P(d \mid v_4)$
d_1	0	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{2}$
d_2	0	$\frac{1}{2}$	$\frac{1}{3}$	0
d_3	1	0	$\frac{1}{3}$	$\frac{1}{2}$

For the given corpus and query terms, we get the following results:

q	$P_2(qv_1)$	$P_2(qv_2)$	$P_2(qv_3)$	$P_2(qv_4)$	$\sum_{j'} P_2(qv'_j)$
q_1	0.002604	0	0.001922	0.002954	0.007480
q_2	0	0.03568	0.02530	0.009452	0.07043

Now compute the $\dfrac{P_2(qv_j)}{\sum_{j'} P_2(qv_{j'})}$ terms:

q	$\frac{P_2(qv_1)}{\sum_{j'} P_2(qv_{j'})}$	$\frac{P_2(qv_j)}{\sum_{j'} P_2(qv_{j'})}$	$\frac{P_2(qv_j)}{\sum_{j'} P_2(qv_{j'})}$	$\frac{P_2(qv_j)}{\sum_{j'} P_2(qv_{j'})}$
q_1	0.3481	0	0.2570	0.3949
q_2	0	0.5066	0.3592	0.1342

Similar to the method 1 calculations above, we now calculate the term weights:

v_j	$\log\left(\frac{P_2(v_j\|r,\vec{q_1})}{P(v_j)}\right)$	$\log\left(\frac{P_2(v_j\|r,\vec{q_2})}{P(v_j)}\right)$
v_1	$\log\left(\frac{0.3481}{\frac{1}{12}}\right) = 0.6209$	$\log\left(\frac{0}{\frac{1}{12}}\right) = -\infty$
v_2	$\log\left(\frac{0}{\frac{17}{45}}\right) = -\infty$	$\log\left(\frac{0.5066}{\frac{17}{45}}\right) = 0.1274$
v_3	$\log\left(\frac{0.2570}{\frac{31}{90}}\right) = -0.1272$	$\log\left(\frac{0.3592}{\frac{31}{90}}\right) = 0.01822$
v_4	$\log\left(\frac{0.3949}{\frac{7}{36}}\right) = 0.3077$	$\log\left(\frac{0.1342}{\frac{7}{36}}\right) = -0.1610$

giving us document scores as follows:

q	$score_q(d_1)$	$score_q(d_2)$	$score_q(d_3)$
q_1	$-0.1272 + 0.3077 = 0.1805$	-0.1272	$0.6209 - 0.1272 + 0.3077 = 0.8041$
q_2	$0.1274 - 0.1610 = -0.0336$	0.1274	-0.1610

In terms of the rankings of the documents with respect to query q_1, we see no difference between the two methods for this particular example: both methods rank d_3 highest, then d_1, and d_2 lowest. Intuitively, this seems like a good ranking with respect to q_1. Recall query q_1 was the string "a super machine". Documents d_3 ("a machine super machine") and d_2 ("big super machine") seem to have content about "super machines". One interesting point is that to a human reader the article "a" in q_1 seems to carry little information, whereas the scoring functions assign the highest weight to "a" (term v_1); however, this is probably explained by the fact that we have a rather unusual corpus where the term "a" has a high IDF, since it only occurs in one document.

With respect to query q_2 ("super big"), the two methods provide very different rankings. Method 1 gives the same ranking for q_2 as for q_1: document d_3 is highest, d_1 comes second, and d_2 is lowest; method 2 reverses the ranking: d_2 is highest, d_1 is second, and d_3 is lowest. In this case, method 2 seems

to perform better, although it can be argued that neither ranking performs optimally. Intuitively, d_1 is probably most relevant to the query, since it contains both query terms. One can argue that document d_2 is more relevant to the query than d_3, on the basis that "super" is being used as an intensifier, so the repetition of "big" in d_2 implicitly carries the meaning of the term "super," whereas document d_3, although it contains "super," does not seem to have any connection with "big." Moreover, "big" seems to be the more important term in the query, since "super" is a modifier applied to "big." Note that method 2 accurately captures this intuition by assigning a higher term-weight to "big" than "super," while method 1 assigns a higher weight to "super" than "big."

Let us examine why this is the case. First consider method 1. Recall the basic procedure to find the probability $P(qv_j)$ in method 1: generate the complete string qv_j with the same document-induced language model, and average the results over all documents. Note that documents that do not contain all query terms cannot generate qv_j; however, they also cannot generate any other $qv_{j'}$, so they do not contribute to the normalization constant $P_1(q)$ either. Therefore, we are only concerned with documents that contain all query terms, and the weight assigned to v_j by method 1 increases if v_j occurs frequently in documents that contain all query terms, so that documents that contain all query terms are likely to generate qv_j.

Now consider method 2. In method 2, the procedure to compute the probability $P(qv_j)$ is essentially as follows: for each query term, compute the average over language models induced by documents containing v_j of the probability of generating the term; now use these probabilities to compute the total probability of generating the query string. Therefore, in method 2, $P(qv_j)$ (and consequently the weight on term v_j) increases if query terms are highly likely to occur in documents that contain term v_j. Note that this is a sort of flipped version of the condition for high term-weights in method 1. The example in this exercise illustrates that the two approaches are in fact different: from the perspective of method 1, document d_1 is the only document to contain all terms of q_2. Since d_1 has a higher than average rate of occurrence of term "super" and lower than average rate of term "big," the term "super" receives a positive weight and "big" a negative weight. From the perspective of method 2, documents d_1 and d_2 contain "big," and those documents have very high rates of occurrence for terms in q_2, giving "big" a large weight; documents d_1 and d_3 contain "super," and those documents have lower rates of occurrence for query terms, giving "super" a smaller weight. Note that method 2 captures the intuition that v_j should receive a high weight if the presence of v_j in document d is evidence of a high frequency of query terms in d, while it is not quite as clear how to interpret a high term-weight in method 1.

References

[LC01] Victor Lavrenko and W. Bruce Croft. Relevance based language models. In *SIGIR '01: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 120–127, New York, NY, USA, 2001. ACM.

[LZ02] John Lafferty and Chengxiang Zhai. Probabilistic relevance models based on document and query generation. In *Language Modeling and Information Retrieval*, pages 1–10. Kluwer Academic Publishers, 2002.

[MP95] David MacKay and Linda Peto. A hierarchical Dirichlet language model. *Natural Language Engineering*, 1(3):289–307, 1995.

[PC98] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. pages 275–281, 1998.

[RJ76] Stephen Robertson and Karen Spärck Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 1976.

[SBM96] Amit Singhal, Chris Buckley, and Mandar Mitra. Pivoted document length normalization. In *SIGIR '96: Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 21–29, New York, NY, USA, 1996. ACM.

[XC00] Jinxi Xu and W. Bruce Croft. Improving the effectiveness of informational retrieval with local context analysis. *ACM Transactions on Information Systems*, 18:79–112, 2000.