Representing and Accessing [Textual] Digital Information (COMS/INFO 630), Spring 2006
Lecture 8: **Completion of alternate LM formulation; introduction to relevance feedback**
2/21/06
Lecturer: Lillian Lee
Scribes: Nicholas Hamatake (nh39), Min Park (mp257)

---

1 Introduction

1.1 Overview

Last lecture, we started examining the language modeling (LM) approach to the probabilistic retrieval model, following previous lectures regarding the Classic approach. The inference scenario for LM retrieval was considered, and a specific example was introduced: *multinomial "topic" models with corpus-dependent Dirichlet priors*.

Today's lecture concludes the discussion of this special case of LM retrieval, including a discussion of an apparent 'problem' in its scoring function, and possible ways to fix it. The suggested ways of fixing the problem were not very satisfactory. This led to a re-examination of this 'problematic' scoring function via some algebraic manipulations that supported the claim that there was no problem to begin with!

Following the conclusion of this model, an important note on vocabulary was made to emphasize that these multinomial models are not equivalent to unigram models. They are similar in some aspects and comparisons are made as well as a statement of their mathematical relationship. Finally, the unigram model, its 'flaw', and solution for it are presented.

The lecture concludes with a new scenario for a re-interpretation of the LM model by which relevance can be explicitly described (previously relevance had been essentially simply a formal hidden variable in our models).

1.2 Review of Probabilistic Retrieval models

1.2.1 Classic Probabilistic Retrieval: scoring by P(R=y|D=d)

Scoring functions for classic probabilistic retrieval were derived from P(*R=y|D=d*) where R and D were used to denote random variables for *relevance* and *document*, respectively. To give P(*R=y|D=*d) a probability distribution of some sort, we assumed that given a document, its relevance status may vary due to document representation. This allowed us to bin documents into attribute classes. For instance, two documents containing two words, "white house" and "house white" could be represented by the same attribute vector, but their relevance status is likely to be different. Finally, making the assumption that attribute variables are binary, the Robertson-Spärck-Jones (RSJ) model ('76) was derived.

For an estimate of the RSJ scoring function that would be a 'derivation of *inverse document frequency',* Croft and Harper ('79), imposed an assumption that non-query term attributes have the same distribution over relevant documents as over all documents, which led to an estimate containing a term equivalent to *idf* in the VSM scoring function. What this estimate was missing with respect to the VSM scoring function was *term frequency* and *length normalization*. That is, with enough assumptions, the derivations from first principles led to the derivation of an *idf*.

A different approach—the 'isolated poisson' model—applied to the RSJ model, would produce *tf* and *idf* in the scoring function together. This approach aimed to model the number of times a particular term occurs in a document of fixed length. With the fixed length assumption, the poisson becomes relevant as the number of times a term occurs among a fixed number of words in a document can be viewed to have a binomial distribution; thus, a poisson approximation of the binomial distribution was used to take terms in

the RSJ model, $P(A_j = a_j(d) \mid R=y)$ and $P(A_j=a_j(d))$ to be probabilities from the poisson distribution with parameters ρ and γ respectively where A_j represents the random variable for attribute vector $a_j(d)$ (note: $\rho > \gamma$). With poisson probabilities substituted into the RSJ model, taking the log of it yields a function containing the equivalent of *tf* and *idf*, but no *length normalization*.

Finally, another approach—the 'mixture of poissons' model—which introduces the notion of *topics* through a set of *m* binary random variables $T_1, \ldots T_j, \ldots T_m$ such that for any *j*, if d if about v(j)'s topic, $T_j = y$, and *n* otherwise. This approach led to a "very big" scoring function with four unknowns once the new random variables were incorporated into the RSJ model. Based on some characteristics of the function, a much more simplified version of the function with similar properties was taken to get a sort of *ad hoc* weighting scheme (BM25/Okapi weighting scheme) that had *tf*, *idf*, and obscurely appended length *normalization* (but still different from VSM).

1.2.2 Language Modeling Approach to the Probabilistic Retrieval, score by P(R=y | D=d, Q=q)

If there was dissatisfaction from the derivations in the classic approach, it could be addressed by rethinking the entire approach. This gave rise to the *language modeling* approach to the probabilistic retrieval model. Lafferty and Zhai ('03) introduced a random variable for the query into the scoring function. This led to the derivation of the scoring function $P(Q=q \mid R=y, D=d)P(R=y \mid D=d)$ where $P(Q=q \mid R=y, D=d)$ was taken as the ranking function, and $P(R=y \mid D=d)$ as a prior probability of relevance. A feasible interpretation of $P(Q=q \mid R=y, D=q)$ was that given a document that satisfies the user's information need, query *q* is issued. In the inference scenario, if $t(d)$ denotes the "topic" of *d*, then $P_{t(d)}(q)$ was used to denote the probability of generating the query *q* if $t(d)$ satisfies the user's information need, whereas in the probabilistic retrieval case, we could think of $P_{t(q)}(d)/P(D=d)$ as incorporating the probability of generating document *d* under the query model $t(q)$. In this LM approach to retrieval, we noticed that there is no hidden variable *R* for relevance. A multinomial model with corpus-dependent Dirichlet priors was introduced as a special case of this inference scenario.

2 Multinomial Topic Models with Corpus-Dependent Dirichlet Priors

2.1 Review and Notation

As a special case of the LM probabilistic retrieval model, the concept of multinomial topic models with corpus-dependent Dirichlet priors was introduced in the previous lecture which ended on an inquisitive note as to how to obtain estimates of $\vec{\theta}$.

The overall idea of this multinomial model is to score a document by $P_d(q)$, the probability that a document model induces the query *q*. We used a multinomial model for generating strings (queries, documents, or an entire corpus). A string of length *l* is produced by *l* independent rolls of a weighted *m*-sided die (where *m* is the size of our vocabulary). This *m*-sided die is represented by a parameter vector $\vec{\theta} = (\theta_1, \ldots, \theta_m)$ such that for all , $j \in [1, m]$,

$\theta_j =$ probability of $v^{(j)}$ appearing

and

$\vec{\theta}(d) = \{$probability of $v^{(j)}$ appearing for a model based on document $d\}_{j=1}^{m}$

For document *d*, $P_{t(d)}$ is the language model represented by $\vec{\theta}(d)$.
Given estimates for the $\vec{\theta}$ parameters, we can express $P_{\vec{\theta}(d)}(q)$ as:

$$P_{\vec{\theta}(d)}(q) = f(q,d)K(q) \tag{2.1.1}$$

Where, *f(q,d)* denotes the probability of seeing the 'sorted' version of *q*.
A sorted query looks like:

$$\underbrace{v^{(1)}\ldots v^{(1)}}_{tf_1(q)\text{ times}} v^{(2)}\ldots v^{(2)}\ldots \underbrace{v^{(m)}\ldots v^{(m)}}_{tf_m(q)\text{ times}}$$

K(*q*) denotes the number of possible permutations of the sorted version of *q;*
it is the multinomial coefficient of our distribution; since each of these coefficients are constants, it will not affect the score in ranking.

Given our (now standard) independence assumption, we were able to express *f*(*q*,*d*) as

$$f(q,d) = \prod_j \left[\theta_j(d)\right]^{tf_j(q)} \tag{2.1.2}$$

2.2 The Anti-idf "Problem"

The estimator proposed for each θ_j was

$$\hat{\theta}_j(d) = \frac{tf_j(d) + \mu tf_j(c)/|c|}{|d| + \mu} \tag{2.2.1}$$

which 'made sense' intuitively, and also provided a *tf* and a *norm* ($|d|$). The $tf_j(c)/|c|$ was added as a smoothing term. This prevents documents not containing all query terms from automatically being given a score of zero. However, we were concerned about having something that looked like an *idf* in the numerator of the scoring function. Recall that the scoring function produced by using the estimates in (2.2.1) is:

$$P_{\vec{\theta}(d)}(q) \overset{rank}{=} f(q,d) = \prod_j \left[\frac{tf_j(d) + \mu tf_j(c)/|c|}{|d| + \mu}\right]^{tf_j(d)} \tag{2.2.2}$$

We called the smoothing term an "anti-*idf*" since it was precisely the reciprocal of what we wanted to see. Using this scoring function, one would think that terms that occur more frequently in the corpus would be given more weight in scoring, which seemed like somewhat of a problem. Possible solutions to this problem that were briefly discussed were as follows:

- Remove the smoothing term (set μ=0)
  Problem with this suggestion: Documents missing any of the query terms are given a score of zero, the same as if they were missing all query terms.
- Smooth by a uniform prior
  While this solution may be mathematically sound, the model would become less sophisticated with respect to language, which seems undesirable.

- Manually replace the $tf_j(c)/|c|$ with its reciprocal, thus forcing an *idf* to appear
  This solution seems far too *ad hoc*.

Alternatively, we could move away from the generative model and start over completely. The point: none of these suggestions really seem to satisfactory address the anti-*idf* issue.

2.3 Was there a problem to begin with?

We now claim that the model really doesn't have the anti-*idf* problem to begin with. We can show this with some algebraic manipulations which are similar in spirit to the ones we saw in the derivation of the Robertson-Spärck-Jones model. In the end, we will have an expression with $tf_j(c)$ in the denominator instead of the numerator.

First, we simplify the expression by defining *norm(d)* to be $(|d|+\mu)^{|q|}$ and moving the denominator outside of the product. We then separate the product into two smaller products, one over terms which appear in the document and one over terms which do not. Finally, we multiply by both the *idf* term and its reciprocal, but only for terms which appear in the document.

Just as in the RSJ model, we get four products. If we denote these as (I) through (IV), the document *tf*'s in (II) are all zero, so (II) and (III) can be combined to form a single term that is constant with respect to *d*, allowing us to drop it from the scoring function. (I) and (IV) can then be combined and simplified to yield our final scoring function:

$$\text{Define } norm(d) = norm(d,\mu,q) = (|d|+\mu)^{|q|}$$

$$P_{\vec{\theta}(d)}(q) \overset{rank}{=} f(\vec{q},d) = \frac{1}{norm(d)} \times \ldots$$

$$\times \prod_{j:tf_j(d)>0} \left[\frac{tf_j(d) + \mu tf_j(c)/|c|}{|d|+\mu} \right]^{tf_j(q)} \quad (\text{I})$$

$$\times \prod_{j:tf_j(d)=0} \left[\frac{tf_j(d) + \mu tf_j(c)/|c|}{|d|+\mu} \right]^{tf_j(q)} \quad (\text{II})$$

$$\times \prod_{j:tf_j(d)>0} \left[\frac{tf_j(c)}{|c|} \right]^{tf_j(q)} \quad (\text{III})$$

$$\times \prod_{j:tf_j(d)>0} \left[\frac{|c|}{tf_j(c)} \right]^{tf_j(q)} \quad (\text{IV})$$

$$\overset{rank}{=} \frac{1}{norm(d)} \prod_{j:f_j(d)>0} \left[\frac{tf_j(d)\,|c|}{\mu tf_j(c)} + 1 \right]^{tf_j(q)}$$

Note that this scoring function contains *length normalization, term frequency*, and an *inverse document frequency*—all three consensus points for the term weighting question (See lecture 2 and 3). Also note that this derivation did not involve any additional independence assumptions or probability estimates; it was accomplished purely algebraically.

3 Multnomial vs. Unigram Models

In the special case of the LM model above, it is important to note that it is different from a unigram model, although there are similarities. In practice, if a unigram model is used for language modeling, one may encounter 'check-sum' issues, such as seemingly appropriate probabilities that actually sum to more than 1.

3.1 The Unigram Model

Assume that the parameters are $\theta_1,\ldots,\theta_m$ (probabilities) as before, such that $\sum_j \theta_j = 1$. Assume independence of term occurrences. Then,

$$"P_u(v^{(j)})" = \theta_j \tag{3.1.1}$$

Since we are assuming independence of term occurrences,

$"P_u(v^{(j)})" = \theta_j$ for all *j,*

and $\sum_j P_u(v^{(j)}) = \sum_j \theta_j = 1$ for all *j*

Then what is $"P_u(v^{(1)}v^{(2)})"$ (as an example)? We might expect it to be equal to $\theta_1\theta_2$ since term occurrences are independent; however, this is a contradiction to $\sum_j \theta_j = 1$ -- all of the available probability has been "used up" by the one-term documents. Thus, $"P_u(v^{(1)}v^{(2)})" = 0$.

The multinomial model differs in its inclusion of a length-dependent term, although it *is* equivalent to the unigram model in terms of ranking (since the length-dependent term depends only on the query and not on the document).

3.2 How to fix 'check-sum' issues in the unigram model

We may still want to use the unigram model's estimates for $\theta_1,\ldots,\theta_m$ for non-ranking purposes. Unlike the multinomial model, there is no computation-heavy coefficient in the unigram model, making it easier to work with.

We can fix the 'check-sum' issues by observing the following: At some point during document generation, a decision must be made to stop generating the document. We can introduce *s* into our model as the probability that this decision is made. This will give us a 2-state HMM model, and $"P_u(v^{(1)}v^{(2)})"$ would now be given by:

$$P_H(v^{(1)}v^{(2)}) = (1-s)\theta_1(1-s)\theta_2 s$$

Note that 1-s is simply the probability of not stopping, the complement of the event of stopping.

It was asserted—but not formally proven—that this gives a proper probability distribution.

4 An even more generative approach

We have produced a VSM-style model as a special case of the language model scenario. But, this derivation finessed selection versus generation, and we may want a model that is a bit more generative.

To this end, we consider an entirely new scenario in which corpus contents are described by a set of "topic" models of language. Documents are generated by choosing a single topic model T_D and using it to generate the document. Queries are similarly generating by selecting a model from a set of language models which describe users' information needs. If we call the query topic model T_Q, we can now express relevance as $T_D = T_Q$. Finally, we are defining relevance explicitly in terms of topic match, rather than treating it as a hidden random variable. We will explore this model more next time...