CS630 Lecture 6: The BM25/Okapi method
Date: February 14th, 2006
Lecturer: Lillian Lee
Scribes: Ari Rabkin and Victoria Krafft

In this lecture, we'll describe BM25/Okapi, a state-of-the-art (classic) probabilistic retrieval approach.

Contents

1 Last Time

Our version of the RSJ model is:

$$\prod_{j:q_j>0,a_j(d)>0} \frac{P(A_j = a_j(d)|R = y)}{P(A_j = a_j(d))} \cdot \frac{P(A_j = 0)}{P(A_j = 0|R = y)}$$

which can be found on the handout
(http://www.cs.cornell.edu/courses/cs630/2006sp/handouts/lec06.pdf) distributed in class. This model offers a paradigm for scoring functions, and we have already discussed two instantiations:

- Using binary A_j attributes and the Croft-Harper assumptions, we got an idf term-weighting scheme. Because the attributes were binary, term frequency did not appear in the scheme.
- We then tried using the *isolated Poisson* model, which was also referred to as the *one Poisson* model. For clarity, all our notes will refer to it as the isolated Poisson model. When combined with some "okay" assumptions, this model gives us a tf-idf scheme.

At this point, all we've done is reproduce vector space results. While it's nice to have a theoretical basis for those results, we would like to improve upon them.

2 A Better Approach

One idea which our model isn't incorporating yet is the notion that documents have subjects. To further examine this idea, fix a term $v^{(j)}$, and assume it corresponds to a topic.

Now, why does term $v^{(j)}$ occur in document d?

There are two possibilities to consider. One is that d is about the topic denoted by $v^{(j)}$. The terminology which is commonly used for this is that "d is *elite* for $v^{(j)}$." The other case is that the term is not related to the topic of d, and just appears by coincidence.

If we're using topics, we probably want to match topics between the document and the query. This is hard to do directly; RSJ has no room for the topic, after all.

So within RSJ, let's introduce a binary random variable, $T_j(d)$. With the document fixed, we'll just write this as T_j. T_j is y ("yes"), if d is about the topic corresponding to $v^{(j)}$, and n ("no") otherwise.

Well, in general,

$$P(B) = \sum_x P(X = x, B)$$

This rule is handy for introducing new random variables; we use it frequently in machine learning.

Applying it in our model, we get:

$$P(A_j = a|R = y) = \sum_{t \in \{y,n\}} P(A_j = a, T_j = t|R = y)$$

Sanity check: did we help ourselves here? Well, the new expression is more sophisticated, but also more complex. Let's be optimistic and press on.

We want to condition on the topic, since term count depends in some sense on the topic. Doing this, we get:

$$P(A_j = a|R = y) = \sum_{t \in \{y,n\}} P(A_j = a|T_j = t, R = y)P(T_j = t|R = y)$$

Note that this now corresponds to a generative model, in which we're assuming that a person picks the topic, and then generates terms according to the topic's term distribution.

Assume that A_j is conditionally independent of R given T_j. That is, if you know the topic,

the relevance doesn't affect word choice. This makes sense, as relevance is defined in terms of the query, which occurs long after the document is written. Note that we didn't say that A_j was independent of R; the topic is tied both to R and A_j.

Now this lets us drop the conditioning on the relevance variable, giving us: .

$$P(A_j = a|R = y) = \sum_{t \in \{y,n\}} P(A_j = a|T_j = t)P(T_j = t|R = y)$$

Now, let $P(A_j = a|T_j = y)$ be a Poisson distribution with parameter $\rho^{(j)}$. $\rho^{(j)}$ is the expected number of occurrences of $v^{(j)}$ for a given document length.

Similarly, let $P(A_j = a|T_j = n)$ be Poisson with parameter $\mu^{(j)}$. For poth of these, we will suppress the dependence on j for clarity, using μ and ρ without superscripts. This gives us the following equations:

$$P(A_j = a|R = y) = \text{Poisson}(a, \mu)P(T_j = n|R = y) + \text{Poisson}(a, \rho)P(T_j = y|R = y)$$

Presumably, μ is smaller than ρ.

So what's not known? We don't have ρ, we don't have μ, we don't have $P(T_j = y|R = y)$, and we also have one more unknown, from expanding $P(A_j = a)$ as a mixture of Poissons.

We would like to get something clever, but alas, no. When we plug into the RSJ model, we get a very ugly function of the a_js (that is, of the term frequencies), among other quantities. This is hard to handle, but it turns out that this function is monotonically increasing to an asymptote, which, if $T_j = y$ and $R = y$ were correlated, would be the idf.

So, let's approximate it! Let's replace that messy term with something with the same limit. Well, for any constant K

$$\lim_{a_j \to \infty} \frac{a_j}{K + a_j} = 1$$

so $\frac{a_j}{K+a_j} \cdot idf$ has the limit behavior we want. This is somewhat magical, but it appears to work in practice.

We still need some sort of length normalization. When we used the Poisson distribution, we were assuming that documents all had the same length. One approach would be to model length as an attribute, but this gets very ugly. Instead, let's use that constant K we have floating around to model document length. In particular, let's use $K = K_1((1-b)+\frac{b \cdot len(d)}{avgdoclen})$[1]. This may look familiar. It's exactly the scheme used in pivoted normalization!

[1]The Singhal table in the handout for the lecture has a typo for this quantity.

This gives us an overall scoring function of:

$$\sum_{t \in Q,D} \ln \frac{N - df + 0.5}{df + 0.5} \cdot \frac{(K_1 + 1)tf}{K_1((1-b) + b\frac{dl}{avg(dl)}) + tf} \cdot \frac{(K_3 + 1)qtf}{K_3 + qtf}$$

Where N is the number of documents in the collection, df is the number of documents containing the term, tf is the frequency of the term's occurence in the document (observe that tf for a term t is equivalent to a_t), qtf is the frequency of the term's occurence in the query, dl is the document length, and $avg(dl)$ is the average document length. K_1, K_3, and b are tuneable constants.

That funny term with the qtf is motivated by wanting to treat query terms like terms in the document, but it's really rather ad hoc. It should make you sad, by the way, that we have to introduce so much ad-hocery in what was supposed to be a probabilistic model. BM25 actually stands for "Best Match, version 25": it took a lot of trial and error to come up with BM25. Despite the various hacks involved, Okapi actually does work well. Most TREC systems actually wound up using some variant of Okapi.

3 A Discussion of Where We Are

Let's step back and evaluate. The BM25/Okapi model is different from the vector space model, which is what we wanted. But it's hard to avoid the feeling that something went wrong. For one thing, it was too hard to use the RSJ model, and so we had to make quite peculiar approximations.

But we might go a step further, and reevaluate our initial assumptions. Suppose for instance that R weren't a hidden variable, but that we could measure it, perhaps because we had some sort of relevance feedback, either explicit or implicit.

"Explicit" feedback would mean that a user gave us query-specific relevance labels. It's hard to get users to do this, however, since it's a real commitment of time and energy, for potentially doubtful return. After all, if the user had relevant documents to evaluate, doesn't that mean that their information need is already satisfied, in which case why do they want to continue interacting with the system? (We'll discuss this more in another lecture.) Note also that it has to be done per query. If you were going to do it, by the way, you might slip in a few documents with comparatively low scores. That way, if the user marks them relevant, you've really learned something.

There's also implicit information about relevance; click-through data is an example of this. There's also examining a user's emails, documents, webpage views and so forth. Obviously, we'd like to do this locally; people are going to be much more comfortable with a local process than with giving all their emails to Microsoft to analyze.

We could also rethink the probabilistic approach. In the basic RSJ schema, the query appears only in the index set; this seems not to be giving it enough prominence, since after all it's the only thing we know about what the user wants.

There's a technical fix, given by Lafferty and Zhai in 2003. Suppose we scored by $P(R = y|D = d, Q = q)$. We need some notion of a document being selected, and of a process creating the query.

We still have the problem we had initially with RSJ, though. It's treating R as a random variable, but in fact, for a given document and query, the relevance is either "yes" or "no". It doesn't really seem like there's a "probability" of relevance involved.

4 Question

a) In the BM25/Okapi formula, which term is the "inverse document frequency"? Which term is the term frequency? Where is the length normalization?

Answer:

The term frequency (with length normalization) is:

$$\frac{(K_1+1)tf}{K_1((1-b)+b\frac{dl}{avg(dl)})+tf}$$

The inverse document frequency is:

$$\ln \frac{N-df+0.5}{df+0.5}$$

The length normalization is:

$$K_1\left((1-b)+b\frac{dl}{avg(dl)}\right)$$

b) Explain what all the constants do in the BM25/Okapi formula. What happens if we raise or lower K_1? b? K_3?

Answer: K_1 controls the influence of higher term frequency. That is, as K_1 grows, the term frequency has a larger impact on the ranking. K_1 is generally chosen between 1

and 2.

The b parameter controls length normalization, and is normally in the range $[0, 1]$. As b grows larger, the system depends more on $\frac{1}{\frac{dl}{avg(dl)}}$, which favors smaller documents. Note that for small b, the length ratio is given very little weight, and thus the larger average tf of longer documents isn't compensated for. b is commonly set to 0.75.

K_3 controls how much weight we give repeated words in the query: As K_3 grows larger, repeated words have more effect. K_3 is usually taken to be between 1 and 1000, with 7 and 1000 being popular values.

c) Observe that while b and K_1 are tuned based on properties of the corpus, K_3 seems to depend only on the queries. The $\frac{(1+K_3)\cdot qtf}{K_3+qtf}$ is essentially weight of evidence that the query is elite for the given term, estimated by the number of occurrences of the term in the query. Since this does not depend on the documents, how do you tune K_3?

Answer: Presumably, you tune it by finding large numbers of "typical" queries. This requires some sort of generative query model or a significant number of queries and corresponding relevance-labeled document sets. Folklore (and a number of published papers) suggest that large values of K_3 work well, and it has been suggested to simply replace the term containing the qtf with the limit as K_3 goes to infinity, which is simply equal to the qtf.

One might wonder how often users reuse keywords in a query. In fact, the actual production Okapi system employed automatic query expansion, in order to allow matching when a document doesn't include the query term precisely. If the query expansion dredged up the same word multiple times, this would be a sign that that keyword was elite for the query, even if not there explicitly. Smaller values of K_3 limit how much extra weight a term gets from being repeatedly suggested by the automatic term expansion.

Note:

BM25 was introduced at TREC-3, and described in :
S. E. Robertson, S. Walker, M. M. Hancock-Beaulieu, and M. Gatford. Okapi at TREC-3. In *Proceedings of the Third Text REtrieval Conference (TREC-3)*, NIST Special Publication 500-225, 1995. http://citeseer.ist.psu.edu/robertson96okapi.html