CS 630 Lecture 5: 02/09/2006

Lecturer: Lillian Lee Scribes: Asif-ul Haque, Benyah Shaparenko

---

1 Review of Probabilistic Retrieval

In **Probabilistic Retrieval** we try to get the probability P(relevant|doc). The approach taken is summarized by the following steps.

1. Convert a document to attributes by binning over some set of terms. So a document d becomes

   $$\overrightarrow{a}(d) = (a_1(d), a_2(d), \ldots, a_m(d))$$

   We have **attribute variables** A_j that receive the values of the attributes.

2. Do the "Bayes flip" to get a different form of the same expression. Here, A is the vector of random variables A_j.

   $$P(R|A) = \frac{P(A|R)P(R)}{P(A)}$$

3. We assume some sort of independence of attributes and factor the conditional probabilities accordingly.

   $$\prod_j \frac{P(A_j = a_j(d)|R = y)}{P(A_j = a_j(d))}$$

   These steps are motivated by estimation. We incorporate the query to

   - get evidence for relevance
   - reduce the number of R-dependent terms so that noise is lower.

We assume A_j corresponds to $v^{(j)}$. We also make the assumption that when $v^{(j)}$ is not in query q the distribution of attribute A_j is independent of relevance.

$$P(A_j = a_j(d)|R = y) = P(A_j = a_j(d))$$

This assumption allows us to write the product as

$$\prod_{j:q_j=1} \frac{P(A_j = a_j(d)|R = y)}{P(A_j = a_j(d))} \text{ for } v^{(j)} \text{ not in } q$$

2 New idea

Let us restrict to $v^{(j)}$ in d, i.e., $a_j(d) > 0$.

$$\prod_{j:q_j=1 \wedge a_j(d)>0} \frac{P(A_j = a_j(d)|R = y)}{P(A_j = a_j(d))} \times \prod_{j:q_j=1 \wedge a_j(d)=0} \frac{P(A_j = a_j(d)|R = y)}{P(A_j = a_j(d))}$$

If we call the first product **I** and the second product **II**, then the above formula is **I** $\times$ **II**. We observe that in the second product $a_j(d) = 0$. So we can write **II** as

$$\prod_{j:q_j=1 \wedge a_j(d)=0} \frac{P(A_j = 0|R = y)}{P(A_j = 0)}$$

Now the dependence of this product on document d is only by the index of the product. In an attempt to remove this dependency, we introduce two product terms that are reciprocal to each other so that their product equals 1. These are numbered as **III** and **IV** respectively.

$$\prod_{j:q_j=1 \wedge a_j(d)>0} \frac{P(A_j = 0|R = y)}{P(A_j = 0)}$$

$$\prod_{j:q_j=1 \wedge a_j(d)>0} \frac{P(A_j = 0)}{P(A_j = 0|R = y)}$$

The product **II** $\times$ **III** is

$$\prod_{j:q_j=1} \text{(stuff not dependent on } d)$$

So it does not affect ranking. Thus the product we want to estimate becomes **I** $\times$ **IV**.

$$\prod_{j:q_j=1 \wedge a_j(d)>0} \frac{P(A_j = a_j(d)|R = y)}{P(A_j = a_j(d))} \cdot \frac{P(A_j = 0)}{P(A_j = 0|R = y)}$$

This is called **Robertson/Spärck Jones (RSJ)** model although strictly speaking, RSJ would have $P(A_j = 0|R = n)$ not $P(A_j = 0)$, etc. This is actually a paradigm rather than a specific model: it is flexible enough to derive different ranking functions by interpreting the requisite quantities differently under different assumptions. This is quite a contrast to the Vector Space Model.

3 One Estimate of the RSJ Model

This estimate is due to Croft and Harper ('79). In the RSJ model, notice that we must have some sort of relevance estimates.

3.1 Estimating the Probabilities

We will assume that the A_j are binary, i.e., simply measuring term presence or absence, not counts. This assumption means that $P(A_j = 1) + P(A_j = 0) = 1$, i.e., we only have to measure one of $P(A_j = 1)$ and $P(A_j = 0)$. (The notation $\hat{P}$ will be used to denote an estimate.) The first quantity we have to measure is $P(A_j = 1)$, which is estimated by:

$$\hat{P}(A_j = 1) = \frac{\text{number of documents containing } v^{(j)}}{\text{number of documents total in } C} \stackrel{\text{denoted}}{=} \frac{N^{(j)}}{N}$$

Since $P(A_j = 1 \mid R = y) + P(A_j = 0 \mid R = y) = 1$, the other quantity that must be estimated is $P(A_j = 1 \mid R = y)$, which is estimated by:

$$\hat{P}(A_j = 1 \mid R = y) = \text{a constant for all } j, \text{ between 0 and 1}$$

Estimating $P(A_j = 1 \mid R = y)$ by a constant is "goofy" because it says that all the query terms have the same distribution over relevant documents, and in fact does not even affect the ranking because it falls out of the equation with $\overset{\text{rank}}{=}$.

3.2 Relation to IDF

At any rate, in the RSJ model we have

$$\prod_{\substack{j:q_j=1 \\ a_j(d)=1}} \frac{\text{a constant}}{1 - \text{a constant}} \cdot \frac{1 - \frac{N^{(j)}}{N}}{\frac{N^{(j)}}{N}} = \frac{N - N^{(j)}}{N^{(j)}}$$

This should look familiar. Since $N^{(j)}$ is most likely small, this expression is approximately $\frac{N}{N^{(j)}}$, i.e., one variant of IDF. In fact, if we take the log of the above formula, we get the more familiar IDF formula

$$\sum_j q_j \times a_j(d) \times \log \frac{N - N^{(j)}}{N^{(j)}}$$

(because $a_j(d)$ and q_j are binary, they can simply be incorporated into the product). It has been claimed that the above is a theoretical derivation of IDF, but you can take it or leave it.

3.3 What about TF?

We postulated in Lecture 1 that one of the three characteristics that any good ranking function must have is TF. So, why does this formula work without containing anything that looks like TF? As it turns out, the formula (even without TF, as the Croft/Harper experiments were done with binary indexing only) worked well in its day because most documents were short – in fact, the TF weight would have typically been either 0 or 1, which is captured by the assumption that the attribute values are binary.

4 Poisson-based Approach

This approach has been quite popular. It originates from Bookstein and Swanson ('74), Harter ('75), and Robertson and Walker ('94).

4.1 Question Regarding Speed Motivation of this Work

Pre-web, the document corpus was generally static, so it made sense to push as much of the processing into the document portion of the method as possible. These calculations could be done offline.

4.2 On to Poisson...

We will use Poisson distributions to model term counts instead of using binary term presence or absence. Each Poisson parameter represents an arrival rate per term, i.e. it models the number of times a particular term occurs in a document of a *given length*. Since the Poisson model is an approximation of the binomial distribution, we can assume the following scenario: a fixed number of

word slots, and then as with the binomial, for each slot we flip a coin to decide whether to put the term in question in. (But the arrival rate is not the same as TF if document length can vary.)

The Poisson distribution is

$$Pois_\mu(X = x) = \frac{\mu^x}{x!}e^{-\mu}$$

The parameter μ actually takes on a per-term value. There are (at least) two nice things about using the Poisson distribution. First, since

$$E\ X = \mu$$

there is a very natural interpretation for the parameter. Secondly, the probability of a count of zero has a very simple form:

$$Pois_\mu(X = 0) = e^{-\mu}.$$

4.3 An Isolated-Poisson Model

There are two Poisson distributions for each term. (Notation: subscripts may be left off.) One Poisson distribution models the arrival rate on the relevant documents, and the other models the arrival rate on the entire set of documents in the corpus. Specifically, for $P(A_j = a_j(d) \mid R = y)$, use $Pois(\rho)$ (where ρ stands for relevant). For $P(A_j = a_j(d))$, use $Pois(\gamma)$ (where γ stands for general). Assume that $\rho > \gamma$.

For each term in the query, the RSJ quantity is:

$$\frac{(\frac{\rho^{a_j(d)}}{(a_j(d))!} \cdot e^{-\rho}) \cdot (e^{-\gamma})}{(\frac{\gamma^{a_j(d)}}{(a_j(d))!} \cdot e^{-\gamma}) \cdot (e^{-\rho})} = (\frac{\rho}{\gamma})^{a_j(d)}$$

Since this was on a per term basis, this is really

$$(\frac{\rho_j}{\gamma_j})^{a_j(d)}$$

After taking the log of the RSJ model, this becomes

$$\sum_{\substack{j:a_j(d)>0 \\ q_j=1}} a_j(d) \log(\frac{\rho_j}{\gamma_j})$$

In this formula, $a_j(d)$ is basically TF, but does $\log(\frac{\rho_j}{\gamma_j})$ resemble IDF? Not really. We can go back to the Croft and Harper assumption and just treat the numerator as a constant. It would be an interesting research exercise to actually compute or estimate what the value of $\log(\frac{\rho_j}{\gamma_j})$ is.

5 Next Time

The previous models are all weird in the sense that people don't first decide on the relevance and then write the document. Next time we will look at a more generative approach by incorporating topics into the model. The good side of this is that the model will be more intuitive. The bad side is that there will be more random variables introduced into the model.

6 Finger Exercise

The purpose of this finger exercise is to explore a concrete example of the RSJ model and how it relates to VSM. Using the Croft-Harper assumption, the formula we derived involves idf. However it is not idf in its original form. If we use the original definition of idf, the formula becomes

$$\prod_j q_j \times a_j(d) \times (idf_j - 1)$$

where $idf_j = \frac{N}{N^{(j)}}$. Let us investigate the difference in ranking if we use idf_j instead of $idf_j - 1$ in the formula.

Problem

Suppose we have a corpus of 100 documents. There are 5 terms "support vector machine", "machine learning", "information retrieval", "artificial intelligence", "bayes net". The number of documents of the corpus that have these terms are $(2, 20, 75, 90, 40)$ respectively.

Suppose there are two documents we consider, $d_1 = \{$"support vector machines", "machine learning", "information retrieval", "artificial intelligence"$\}$ and $d_2 = \{$"support vector machine", "bayes net"$\}$.

Suppose we have the query $q = \{$"support vector machine", "information retrieval", "artificial intelligence"$\}$.

1. Compute the idf for each term.
2. Compute the scores of the two documents using the formula provided above. Which document is chosen?
3. Compute the scores using idf instead of $idf - 1$ in the formula. Which document is chosen now?
4. Is there any difference in using the two forms of the formula? Explain.

Solution

1. The idf for the terms are $(50, 5, 1.33, 1.11, 2.5)$ respectively.
2. $\text{Score}(d_1) = (50 - 1) \times (1.33 - 1) \times (1.11 - 1) = 1.7787$
   $\text{Score}(d_2) = (50 - 1) = 49$
   So document d_2 is chosen.
3. $\text{Score}(d_1) = (50) \times (1.33) \times (1.11) = 73.815$
   $\text{Score}(d_2) = (50) = 50$
   So document d_1 is chosen.
4. By inspection we see that indeed d_1 is more relevant to the query q since it has more overlapping terms. But the given formula ranks document d_2 higher since when the idf of a term falls below the value 2, i.e., more than half the documents have the term, $idf - 1$ falls below 1. (Indeed, it has been observed that the form has some anomalous behavior.) So these values not only

produce a lower score by their own low value but also diminish the effect of some of the other terms that have higher *idf*.