CS 630 Lecture 10: 02/28/2006

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

---

1 Relevance Feedback Methods

For the purpose of this lecture, let us assume that Relevance Feedback (abbreviated RF) is available in the form of explicit binary (yes/no) judgments for each of the top-k retrieved documents. Although RF can potentially be used in any setting, it is most natural in the probabilistic retrieval model since there is a notion of relevance in the form of a random variable. So let us first discuss how we can use feedback in the probabilistic case.

2 Using RF in the Probabilisitic Model

In the model, we have the RSJ "weight" for each term, assuming binary attribute variables.

$$RSJ_j = \frac{P(A_j = 1|R = y)}{P(A_j = 1)} \frac{P(A_j = 0)}{P(A_j = 0|R = y)}$$

First, we rank the documents. Once we get the relevance feedback, we can use the positive feedback, i.e. documents judged relevant, to estimate the terms $P(A_j = 1|R = y)$ (the quantity $P(A_j = 0|R = y)$ is simply $1 - P(A_j = 1|R = y)$). Then we rerank the documents using the new term weights. One issue that pops up in our mind is what to do with the negative feedback, i.e. documents that were judged not relevant. Here, we note that the original form of the RSJ weight was an "odds ratio" version which had $P(A_j = 1|R = n)$ instead of $P(A_j = 1)$. So we can use the negative feedback if we use this form of the formula. Since the size of the set of irrelevant documents may be much larger than the size of the set of relevant documents as provided by the feedback, we can also think about using only the top-k negative feedbacks. On the other hand, in high precision systems, there will be almost no negative feedback. In practice, the general collection is used for estimation.

3 Reranking with RF

The process of reranking works as follows.

Initially use the Croft-Harper estimate or use query q as "relevant doc"
Produce ranking
repeat
  Get relevance feedback
  Rerank with new RF information

4 Term weighting and Automatic Query Expansion (AQE)

Suppose we have a document containing the word "autos" while the query has the word "cars." Obviously the document is relevant to the query. We can use RF to know that "autos" is relevant

to "cars" so that ultimately the system retrieves the document containing "autos" as a relevant document. But there is a problem here. The RSJ scoring function is

$$score(d) = \prod_{j:v^{(j)} \text{ is in } d \text{ and } q} RSJ_j$$

Here the terms "autos" and "cars" would not appear in the product since none of the words appear in both the document and the query. The solution to this problem is **Automatic Query Expansion (AQE)**. In AQE, we add some more (distinguishing) terms to the query (without the user knowing it) from the relevance information we have. So once the user indicates (indirectly through RF) to the system that "cars" is relevant to "autos," we can add the term "autos" to the query. Robertson does this AQE by first ranking the terms by the scoring function (for terms).

$$selectscore(v^{(j)}) = RSJ_j \times [P(A_j = 1|R = y) - P(A_j = 1)]$$

After ranking the terms, we select the top ranking terms to be added to the query. Then we rank the documents according to the usual scoring function $score(d)$.

4.1 Digging deeper into AQE

One question that is important to understand here is whether the bracketed term $[P(A_j = 1|R = y) - P(A_j = 1)]$ is redundant in the term selection function $selectscore(v^{(j)})$. In other words, we are asking whether we choose different terms to add if we use $selectscore(v^{(j)})$ vs RSJ_j as term selection function. Let us investigate.

Let v and v' be two terms. Let the probabilities associated with these two terms be

$$v: \quad p = P(A = 1|R = y) \quad q = P(A = 1)$$

$$v': \quad p' = P(A' = 1|R = y) \quad q' = P(A' = 1)$$

$$RSJ = \frac{\frac{p}{1-p}}{\frac{q}{1-q}}$$

$$RSJ' = \frac{\frac{p'}{1-p'}}{\frac{q'}{1-q'}}$$

Suppose $RSJ < RSJ'$. So the term v' is preferred by RSJ_j ranking. We want to know if the ranking of the terms by $selectscore(v^{(j)})$ can give a different result. This can happen only when $\frac{p}{q} < \frac{p'}{q'}$. In particular we want to know if $RSJ \times (p - q) > RSJ' \times (p' - q')$ is possible. This can happen only when $(p - q) > (p' - q')$. Suppose

$$p = \frac{1}{2} \quad q = \frac{1}{4}$$

$$p' = \frac{7}{100} \quad q' = \frac{1}{100}$$

Now

$$\frac{p}{q} = 2 < 7 = \frac{p'}{q'}$$

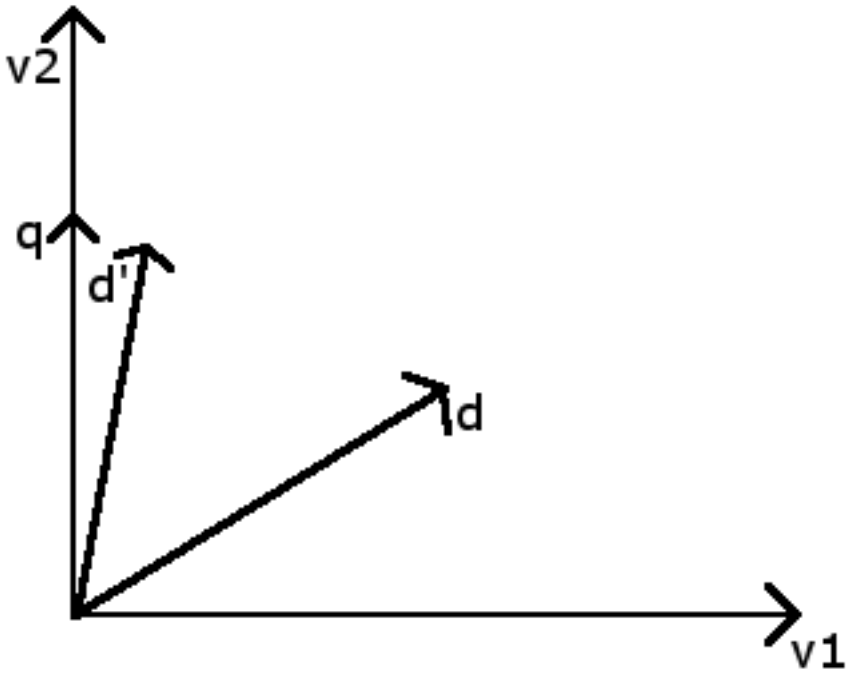
$$p - q = \frac{1}{4} > \frac{6}{100} = p' - q'$$

So in this case the rankings do differ. From this investigation we realize that the RSJ scoring takes into account only relative difference i.e. $\frac{p}{q}$, while the term ranking process also takes into account the absolute difference $p-q$. To get a large absolute difference, a term requires high frequency in the relevant documents and frequent terms may be more reliable. This absolute difference is also more stable than relative difference since ratios may change dramatically by a small change in values. (Of course, why do we not then use this absolute difference for the "actual" scoring of the documents?)

5 Relevance Feedback for the VSM

In the VSM case, RF is also used for query expansion. This work comes from Rocchio '71, Ide and Salton '71, and others.

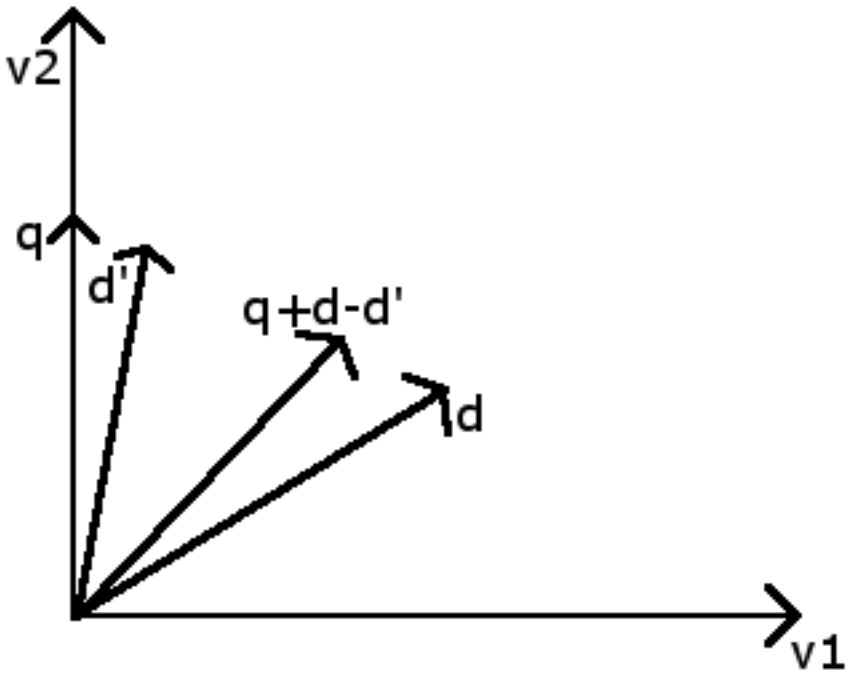
Consider an example: there are two dimensions corresponding to $v^{(1)}$ and $v^{(2)}$, a query q in the direction of $v^{(2)}$, a document d' originally (incorrectly) ranked as relevant, and a document d originally (incorrectly) ranked as irrelevant. This scenario is depicted in this diagram.



The VSM ranks d' above d. To fix this with RF, we perform some simple vector arithmetic. Since document d is relevant, we add $\vec{d}$ to $\vec{q}$. And, since document d' is not relevant, we subtract $\vec{d'}$ from $\vec{q}$. Define the new query to be $\vec{q}_{new}$.

$$\vec{q}_{new} = \vec{q} + \vec{d} - \vec{d'}$$

This scenario is depicted in the following diagram.



More generally, we have the following definition of $\vec{q}_{new}$.

$$\vec{q}_{new} = \alpha\vec{q} + \frac{\beta}{\#\text{ judged relevant}} \sum_{d^{(i)} \text{ judged relevant}} \vec{d}^{(i)} - \frac{\gamma}{\#\text{ judged irrelevant}} \sum_{d^{(i)} \text{ judged irrelevant}} \vec{d}^{(i)}$$

It is possible that $\vec{q}_{new}$ contains negative components, which are sometimes zeroed out. However, negative term weights could be interpreted as an interest in documents without the negative-weighted terms.

The averaging with α, β, and γ is used to handle class imbalance bias. Typically, α, β, and γ are left as free parameters. The tradeoff between α and (β, γ) represents the tradeoff between weighting the query (the user knew what s/he was talking about at the start) and weighting the feedback (the feedback is of better quality than the query). Additionally, if we only want positive feedback, setting $\gamma = 0$ ignores negative feedback. Finally, to make the RF for VSM method work in practice, we need to use some kind of term filtering.

6 Relevance Feedback for Language Modeling

In our first derivation, we had $P(Q = q|D = d, R = y)P(R = y|D = d)$. The $D = d$ in the first quantity is problematic because the relevance feedback is for the document d. For the second quantity, we can use a Bayes flip.

In the second derivation, we defined $(R = y) \equiv (T_D = T_Q)$. There is no (obvious) use of relevance information for the document in question. However, we can use RF to update (the estimate of) T_Q.

This is more sensible if we are ranking by the distance between T_D and T_Q.

Aside : compare this to the "Relevance model" by Lavrenko and Croft '01, which looks at probabilities of the form $P(W|T_R)$, where T_R is an LM that generates relevant documents (and the query).

7 Further Issues

There are several issues with RF. First of all, RF is typically only useful on a per-query basis, i.e., it only makes sense for a particular query at a particular time. Also, automatic query expansion (AQE) is not readily apparent to the user. Terms are added to the query (possibly) without the user's knowledge, so the process is not transparent. For example, searching the Internet (via Google) with the query "Lilian Lee" (three L's total) yields the home page of "Lillian Lee" (four L's total). While this example could result from simply (overzealous?) spelling correction, it can also illustrate the unexpected results from expanding the query by adding the word "Lillian" to the original query.

8 Finger Exercise

1. Suppose we have 5 documents
   d_1 = "apple computers releases new laptop"
   d_2 = "cortland apple is wonderful for salad"
   d_3 = "eat salad stay healthy"
   d_4 = "some irrelevant text"
   d_5 = "more garbage"
   Suppose the terms we consider are {"apple", "computers", "salad", "healthy"}. Let us consider a query q = "apple".

   (a) Rank the documents using probabilistic retrieval using the Croft-Harper assumption.

   (b) If the user identifies the second and third documents (d_2, d_3) as relevant ("apple" as a fruit), which term (if any) will be included in the query using Robertson's term ranking method? Rerank the documents if required. Assume the policy is to add one more term if the term is clearly ranked higher than the other terms.

   (c) If the user identifies document d_3 (instead of d_2 and d_3) as relevant, what will be the effect of expanding the query as above?

2. In AQE, Robertson's proposed reranking model derives a *selectscore* term to determine which words to add to the query automatically. Usually a certain k number of words are added. What effect does selection of k have on the ranking?

3. In the Rocchio method for handling RF in the VSM, the parameters α, β, and γ were used to weight quantities in the new query.

$$\vec{q}_{new} = \alpha\vec{q} + \frac{\beta}{\#\text{ judged relevant}} \sum_{\substack{d^{(i)} \text{ judged} \\ \text{relevant}}} \vec{d}^{(i)} - \frac{\gamma}{\#\text{ judged irrelevant}} \sum_{\substack{d^{(i)} \text{ judged} \\ \text{irrelevant}}} \vec{d}^{(i)}$$

   Make arguments for the following scenarios:

(a) $\alpha > \beta, \gamma$
(b) $\beta, \gamma > \alpha$

Solution

1. The RSJ scoring function for a document is the product of the term weights RSJ_j such that the term $v^{(j)}$ appears in the query and in the document.

$$RSJ_j = \frac{P(A_j = 1|R = y)}{P(A_j = 1)} \frac{P(A_j = 0)}{P(A_j = 0|R = y)} = \frac{P(A_j = 1|R = y)}{1 - P(A_j = 1|R = y)} \frac{1 - P(A_j = 1)}{P(A_j = 1)}$$

Using the Croft-Harper assumption

$$RSJ_j \propto \frac{N - N^{(j)}}{N^{(j)}}$$

where N is total number of documents in corpus, $N^{(j)}$ is the number of documents having the term $v^{(j)}$.

(a) Using the Croft-Harper assumption, we score the documents. Only the term "apple" is important since it is the only word in q.

$$score(d_1) \propto \frac{5-2}{2} = \frac{3}{2}$$
$$score(d_2) \propto \frac{5-2}{2} = \frac{3}{2}$$
$$score(d_3) = score(d_4) = score(d_5) = 0$$

(b) The *selectscore* values for the terms are as follows.

$$selectscore(\text{“apple”}) = \frac{1}{1} \times \frac{3}{2} \times [\frac{1}{2} - \frac{2}{5}] = 0.15$$
$$selectscore(\text{“computers”}) = \frac{0}{1} \times \frac{4}{1} \times [\frac{0}{2} - \frac{1}{5}] = 0$$
$$selectscore(\text{“salad”}) = \frac{1}{0} \times \frac{3}{2} \times [\frac{2}{2} - \frac{2}{5}] = \infty$$
$$selectscore(\text{“healthy”}) = \frac{1}{1} \times \frac{4}{1} \times [\frac{0}{2} - \frac{1}{5}] = -0.8$$

To avoid dividing by 0, we can use the (probabilistically valid) smoothing methods we have considered so far e.g. add δ to the numerator and compensatory normalization to the denominator. The term "salad" is ranked highest by the term ranking method and we add it to the query. The scores of the documents with this expanded query are as follows.

$$score(d_1) = \frac{3}{2}$$
$$score(d_2) = \frac{3}{2} \times \infty = \infty$$
$$score(d_3) = \infty$$
$$score(d_4) = score(d_5) = 0$$

So the documents d_2 and d_3 are ranked highest with the relevance feedback (no surprise).

(c) The values for term and document ranking are as follows.

$$selectscore(\text{“apple”}) = \frac{0}{1} \times \frac{3}{2} \times [\frac{0}{1} - \frac{2}{5}] = 0$$

$$selectscore(\text{“computers”}) = \frac{0}{1} \times \frac{4}{1} \times [\frac{0}{1} - \frac{1}{5}] = 0$$

$$selectscore(\text{“salad”}) = \frac{1}{0} \times \frac{3}{2} \times [\frac{1}{1} - \frac{2}{5}] = \infty$$

$$selectscore(\text{“healthy”}) = \frac{1}{0} \times \frac{4}{1} \times [\frac{1}{1} - \frac{1}{5}] = \infty$$

The terms “salad” and “healthy” are both ranked highly . So we either take them both or decide not to expand the query. Let us consider the case when we expand the query as “apple salad healthy”.

$$score(d_1) = 0$$

$$score(d_2) = 0 \times \infty = 0$$

$$score(d_3) = \infty \times \infty = \infty$$

$$score(d_4) = score(d_5) = 0$$

Now the document d_3 is ranked highest (indicating the user's need of information on diet and health rather than technology) although d_3 does not contain the original query term “apple”!

2. The selection of k has the following effects on the ranking. If k is set too low, then the query may already contain these terms. Or, there may not be enough overlap between these added terms and the terminology of the true set of relevant documents. On the other hand, if k is set too high, then many terms will be added, even ones that may contribute nothing but noise to the ranking process. Additionally, if k is set too high, then erroneous terms may be added to the query, resulting in the system selecting documents that appear to be relevant (to the added query terms, but not the original ones), although the documents are actually not relevant to the user's information need. So would a thresholding scheme make more sense?

3. (a) Of many possible arguments, here are several. The parameter α controls how much weight is given to the user's original query. If the user is skilled in writing queries, then the original query captures much of the user's information need. The relevance feedback may be useful for fine-tuning, but the user knew what s/he was talking about in the query, right from the beginning. Additionally, if $\alpha < \beta, \gamma$, the RF may over-power the original query, resulting in unexpected documents returned after the system incorporates the RF.

   (b) The average user is not skilled at writing queries, but quite good at looking at a document and seeing if it fulfills the information need. By discounting the original query and focusing more on the relevance feedback, the system is using much more accurate information to rank the documents. Even if the user is skilled at writing queries, because of polysemy of the query terms, documents addressing the most popular of the several possible meanings are returned the first time. However, after incorporating RF, with relatively high weights on the RF, a “niche” topic could move to the top of the rankings.