Lecture 5: Introduction to (Robertson/Spärck Jones) Probabilistic Retrieval

Scribes: Ellis Weng, Andrew Owens

February 11, 2010

1 Introduction

In this lecture, we will introduce our second paradigm for document retrieval: probabilistic retrieval. We will focus on Roberston and Spärck Jones' 1976 version, presented in the paper *Relevance Weighting of Search Terms*[1]. This was an influential paper that was published when the Vector Space Model was first being developed — it is important to keep in mind the differences and similarities between these two models and the motivations for each.

Recall that the Vector Space Model was originally a representation model. The retrieval scheme of the Vector Space Model was empirically-driven and chosen in a fairly atheoretical manner. In contrast, probabilistic retrieval is more principled and theoretically-driven. On the other hand, many statistical estimations and empirical substitutions will drive the derivation of this paradigm.

2 Notation

2.1 Problem

First, let us assume a binary label set, L, where the documents are either relevant, r, or not relevant, $\overline{r}$. (The binary distinction is not so important at this time but simplifies presentation.)

$$L = \{r, \overline{r}\}(\text{relevant/irrelevant}) \tag{1}$$

The main goal of probabilistic retrieval is to rank documents by the probability that they are relevant. Intuitively, this can be represented in the following way:

$$\Pr(r|d, q), \tag{2}$$

where d is the document and q is the query. It is important to note that the score of a document ought to be a probability between 0 and 1 (inclusive), not a simple binary relevance score of 0 or 1, for this to be a meaningful scoring function.

2.2 Uncertainty

Question: Why is this not 0 or 1? Shouldn't a document just be either relevant or irrelevant? Answer: There is uncertainty associated with probabilistic retrieval. Uncertainty can arise from any of the following:

1. **Uncertainty with respect to a particular user**.

   A user's judgment for a document relevancy might change from time to time depending on context. The resulting sample space for this uncertainty is

$$Q \times D \times L \times F, \tag{3}$$

where Q is the set of all possible queries (or information need), D is the set of all possible documents, L is the label set, and F is the set of all other factors that might change your threshold.

2. **Uncertainty with respect to different users.**

   For a single document-query pair, there is variation in determining if the document is relevant or not among different users. Thus, we have to take different users into account. The underlying sample space for this uncertainty is

$$Q \times D \times L \times U, \tag{4}$$

   where U is the set of all possible users.

3. **Uncertainty with respect to "lossy" input representation.**

   Depending on the representation, a document can either be relevant or not. It is impossible to take into account all the features of a document. Thus, several document-query pairs yield the same representation; however, these different documents do not necessarily have to be both relevant or not relevant. The resulting sample space for this uncertainty is still

$$Q \times D \times L, \tag{5}$$

   but the "observables" induced from this are different.

4. **Uncertainty with respect to to system uncertainty.**

   The retrieval system itself may be a source of uncertainty. For example, the system might trade off accuracy for speed by using an approximation algorithm or run on a network that is prone to errors.

Question: Why is the type of uncertainty important? Answer: The type of uncertainty presumably affects the estimation and derivation of the probabilistic model.

The ultimate goal of document retrieval is to find documents that are relevant for a different users. One can imagine document retrieval systems that were tailored to a specific user's needs and moods (thus eliminating type-a uncertainty) or document retrieval systems that captured all the important features of the documents and queries (thus eliminating type-c uncertainty); however, document retrieval systems are typically used by more than one user. Solving the problem with respect to b-type uncertainty thus seems to be the goal. Unfortunately, this problem has the worst sample space and seemingly requires extensive user annotation by many users. Because of these problems, our derivation will be based on representational uncertainty (type-c uncertainty).

(Note: Throughout the derivation, it is worthwhile to ask how this model can apply to these other types of uncertainty.)

2.3 Representation

Assume we have a feature-vector representation (which is not implied by any principles we have mentioned so far). A document can be represented as a set of values of m feature functions. A feature function $f_j : D \to \mathbb{R}$ describes the important features of the document. These functions are based on attribute frequency, so 0 must indicate the absence of a particular feature. These feature functions are analogous to the term counts in the VSM, but unlike term counts, these feature functions can, in principle, take more into account than simply the number of times words appear. For example:

$$f_{14}(d) = \text{the number of times "juice" appears in the document} \tag{6}$$

$$f_{16}(d) = \begin{cases} 1 & \text{if } d \text{ contains "Cornell" and "best"} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

$$\vec{f_d} = \begin{pmatrix} f_1(d) \\ \vdots \\ f_2(d) \\ \vdots \\ f_m(d) \end{pmatrix} \tag{8}$$

Similarly, query can be represented as a document:

$$\vec{q} = \begin{pmatrix} f_1(q) \\ \vdots \\ f_2(q) \\ \vdots \\ f_m(q) \end{pmatrix} \tag{9}$$

Note that we will sometimes use the notation $f_d[j]$ for $f_j(d)$ for readability.

Let $\vec{F} = (F[1], F[2], ..., F[m])^T$ be a feature vector. We can write $\vec{F} = \vec{f_d}$ to mean that a feature vector matches the description for document d. Under this notation, we can rewrite our probabilistic scoring function for a given query and a given document d as

$$\Pr(r|\vec{F} = \vec{f_d}, \vec{q}) \tag{10}$$

2.4 Comparison of Scoring Functions

Probabilistic Retrieval	**Vector Space Model**
$\Pr(r\|\vec{F} = \vec{f_d}, \vec{q})$	$\vec{d} \cdot \vec{q}$
Goal-driven. The goal of a probabilistic retrieval model is clearly to retrieve the documents with the highest probability of relevance to the given query.	**More Arbitrary.** The inner product makes intuitive sense, but we have no further justification for this besides empirical evidence.
Lack of Data. There seemingly needs to be data on what documents are relevant (to which queries) in order to compute these probabilities. There can potentially be no data in regards to relevance.	**Computable.** No additional information is needed in order to compute these values, assuming a typical representation scheme.

3 Computation

Assume the query $\vec{q}$ is fixed. We want to compute $\Pr(r|\vec{F} = \vec{f_d}, \vec{q})$ for each document. There are many challenges to this computation. One problem is that relevance is unlabeled in our dataset, so we do not know which documents are relevant for a given query. Another problem is the sparse data for $\vec{F} = \vec{f_d}$. There may be very few documents d and d' such that $\vec{f_d} = \vec{f_{d'}}$.

To deal with the first problem, we will use the "strategy of wishful thinking," a surprisingly powerful principle. We will condition on r, as though we had the full labeled dataset!

First, we present some probability background material. If we have random variables a and b, we can apply the *Bayes Flip*

$$\Pr(a|b) = \frac{\Pr(a,b)}{\Pr(b)} = \frac{\Pr(b|a)\Pr(a)}{\Pr(b)} \tag{11}$$

Terminology: $\Pr(a,b)$ is the *joint* probability of a and b, and $\Pr(b)$ is the *marginal probability* of b.

Applying the Bayes flip, we get

$$\frac{\Pr(\vec{F} = \vec{f_d}|r,\vec{q})\Pr(r|\vec{q})}{\Pr(\vec{F_d} = \vec{f_d}|\vec{q})} \tag{12}$$

Thankfully, we can simplify this equation. $\Pr(r|\vec{q})$ is the probability that a randomly-chosen document is relevant to the user, given his or her query. Since this value is independent of $\vec{f_d}$, the equation is equivalent under ranking to

$$\frac{\Pr(\vec{F} = \vec{f_d}|r,\vec{q})}{\Pr(\vec{F} = \vec{f_d}|\vec{q})} \tag{13}$$

Note that $\vec{F} = \vec{f_d}$ is independent of the the query for systems where the document's representation is not affected by the user's query. So, we can rewrite the above as

$$\frac{\Pr(\vec{F} = \vec{f_d}|r,\vec{q})}{\Pr(\vec{F} = \vec{f_d})} \tag{14}$$

Now consider the denominator $\Pr(\vec{F} = \vec{f_d})$. This is the probability that a randomly-chosen document is represented as f_d in our retrieval system, given the query. We could estimate this probability by counting the fraction of our documents in our database that are represented as f_d. Unfortunately, our data is sparse, so almost always $\Pr(\vec{F} = \vec{f_d}) = 1/n$ (where n is the number of documents) because it is unlikely two documents have the same representation.

To deal with this issue, we assume (as in Cooper[2]) that there exists a constant $k > 0$ such that $\Pr(\vec{F} = \vec{f}) = k\prod_j \Pr(\vec{F}[j] = \vec{f_d}[j])$. If $k = 1$, then we are assuming that each $\Pr(\vec{F}[j] = \vec{f_d}[j])$ is an independent event.

We run into data sparsity issues in the numerator as well, so we make a similar assumption. We assume there is a $k' > 0$ such that $\Pr(\vec{F} = \vec{f_d}|r,\vec{q}) = k'\prod_j \Pr(\vec{F}[j] = \vec{f_d}[j]|r,\vec{q})$. If $k' = 1$ then this is a Naive Bayes assumption. (Aside: we can avoid making one of the assumptions if we start with a slightly more complex scoring function.)

Under these assumptions, we can rewrite the above as

$$\prod_j \frac{\Pr(\vec{F}[j] = \vec{f_d}[j]|r,\vec{q})}{\Pr(\vec{F}[j] = \vec{f_d}[j])}\frac{k'}{k} \tag{15}$$

Note that we can remove the constant $\frac{k'}{k}$ and get an equation that is ranking-equivalent to this one. At this point, we still have no information on the relevance of the documents. Now we try to simplify the equation further by looking at the query, since the terms in the query are the only information we have regarding relevance. We distinguish between terms that appear in the query and those that don't.

$$\prod_{j:q[j]\neq 0} \frac{\Pr(\vec{F}[j] = \vec{f_d}[j]|r,\vec{q})}{\Pr(\vec{F}[j] = \vec{f_d}[j])} \prod_{j:q[j]=0} \frac{\Pr(\vec{F}[j] = \vec{f_d}[j]|r,\vec{q})}{\Pr(\vec{F}[j] = \vec{f_d}[j])} \tag{16}$$

Let us compare the numerator and denominator of the second product. The numerator represents the probability that a *relevant* document has a certain feature value for a feature that is not in the query, while the denominator represents the probability that *any* document has a certain value for a feature that is not in the query. One can argue that these two are similar because words that are left out of a query are usually not

intentionally left out by a user. For example, when a user searches for the word "computer", the user is not intentionally leaving out the words "PC", "Mac", "Desktop", etc. It is impossible to list all the terms that are of interest to a user, so the words that are left out do not necessarily indicate that the user thinks they are irrelevant. So, we don't know whether these terms occur more in relevant or in general documents. Also in large corpora, factoring in the features that are not in a query is not practical, because the vocabulary or set of features of all the documents is massive. Thus, one can make an assumption that the words that are not present in the query are distributed across relevant documents the same as they are distributed across the whole corpus, since we lack any other information to the contrary. This yields:

$$\prod_{j:q[j]\neq 0} \frac{\Pr(\vec{F}[j] = \vec{f}_d[j]|r, \vec{q})}{\Pr(\vec{F}[j] = \vec{f}_d[j])} \tag{17}$$

Although this equation does not seem better than the original probablistic model, there are many terms that were eliminated using assumptions. In the following lectures, we will see the benefits of having the equation in this form.

4 Questions

Recall that the derivation of the probabilistic model required a number of assumptions and independence arguments. This exercise will help you understand the meaning of the assumptions and how each of these assumptions affects the relevance scores of the documents.

Suppose we have a set of 10 documents and 5 queries. Each document can have multiple representations as a vector of feature functions. Assume that the feature functions are binary word counts (the first feature function is whether the word "car" is in the document). Therefore, each document can be represented as a vector of 30 feature functions (there are 30 words in this corpus, so there will be one feature function for each word). The queries can also be represented as documents (also a vector of 30 feature functions). The following table shows the documents (rows), queries (columns), and whether a document is relevant with respect to the query (1 or 0). In practice, we would not have these labels, so it would not be possible to compute these probabilities. In future lectures we will discuss ways to deal with this problem.

	car	toyota	park	green car low milage	toyota brand car
this red car is fast and new toyota is a good brand of car	1	1	0	0	1
this automobile is red and the brand is toyota it is not very fast but it is cheap	1	1	0	0	1
the green car is fast and the mileage is low	1	0	0	1	0
the car with low mileage is under a tree	1	0	0	0	0
the green tree is in ithe park	0	0	1	0	0
the automobile is green and this car is cheap	1	0	0	0	0
park the car under the tree	1	0	1	0	0
the toyota has low mileage and is very cheap	1	1	0	0	0
the car is not a good brand	1	0	0	0	0
there is a automobile that is red and in the park	1	0	1	0	0

1. Using the provided table, estimate the relevance of the first document in the first row with the query "toyota brand car", using the final equation presented in this lecture:

$$\prod_{j:q[j]\neq 0} \frac{\Pr(\vec{F}[j] = \vec{f}_d[j]|r, \vec{q})}{\Pr(\vec{F}[j] = \vec{f}_d[j])}$$

2. Estimate the relevance of the first document using the equation without the assumption that the only significant features are those in the query (i.e. use equation (16) instead of equation (17)).

$$\prod_{j:q[j]\neq 0} \frac{\Pr(\vec{F}[j] = \vec{f}_d[j]|r, \vec{q})}{\Pr(\vec{F}[j] = \vec{f}_d[j])} \prod_{j:q[j]=0} \frac{\Pr(\vec{F}[j] = \vec{f}_d[j]|r, \vec{q})}{\Pr(\vec{F}[j] = \vec{f}_d[j])}$$

Feel free to use a computer program.

3. Compare and contrast the following two tables. An entry is a relevance score for a given (document, query) pair. In the first table, the only features considered were those in the query (as in the first exercise), while the second table uses all of the features (as in the second exercise).

	car	toyota	park	green car low milage	toyota brand car
this red car is fast and new toyota is a good brand of car	1.11	3.33	0	0	Answer 1
this automobile is red and the brand is toyota it is not very fast but it is cheap	0.83	3.33	0	0	13.89
the green car is fast and the mileage is low	1.11	0	0	61.73	0
the car with low mileage is under a tree	1.11	0	0	0	0
the green tree is in ithe park	0.83	0	3.33	0	0
the automobile is green and this car is cheap	1.11	0	0	0	0
park the car under the tree	1.11	0	3.33	0	0
the toyota has low mileage and is very cheap	0.83	3.33	0	0	0
the car is not a good brand	1.11	0	0	0	0
there is a automobile that is red and in the park	0.83	0	3.33	0	0

	car	toyota	park	green car low milage	toyota brand car
this red car is fast and new toyota is a good brand of car	3.74	468.17	0	0	Answer 2
this automobile is red and the brand is toyota it is not very fast but it is cheap	3.18	40782.92	0	0	353160.66
the green car is fast and the mileage is low	1.08	0	0	136672.91	0
the car with low mileage is under a tree	0.85	0	0	0	0
the green tree is in ithe park	0.07	0	2031.53	0	0
the automobile is green and this car is cheap	0.92	0	0	0	0
park the car under the tree	0.35	0	652.99	0	0
the toyota has low mileage and is very cheap	1.45	26.01	0	0	0
the car is not a good brand	1.60	0	0	0	0
there is a automobile that is red and in the park	0.42	0	2571.15	0	0

4. Why do the tables contain values greater than 1?

5. One problem with the approach we have described is that every time a user performs a query, we must check whether each document is relevant. This approach does not scale well. It would be nice if we could process a small subset of the documents instead. We assumed that the features in the query are the only features that influence the probability calculation. Let's make an additional assumption in this same spirit: for any relevant document d, there is a feature j such that $f_j(d) \neq 0$ and $f_j(q) \neq 0$. Is this assumption reasonable? Explain how you could build a more efficient information retrieval system that only looked at a subset of the documents for a given query.

5 Answers

1. This equation is only concerned with words that are in the query, so we only have to take the product of the probabilities of three words. For the word "toyota", the probability of seeing the word "toyota" in the relevant documents is 1. The probability of seeing the word "toyota" in this corpus is 3/10. The word "toyota" contributes 10/3 to the product (note that this is greater than 1). For the word "brand", the probability of seeing the word "brand" in the relevant documents is 1. The probability of seeing the word "brand" in this corpus is 3/10. The word "brand" contributes 10/3 to the product. For the word "car", the probability of seeing the word "car" in the relevant documents is 1/2. The probability of seeing the word "car" in this corpus is 6/10. The word "car" contributes 10/12 to the product. Multiplying these terms yields 9.26.

2. We used a computer program to compute the result: 64866.24. The source code for this program is included at the end of this document.

3. The scores of the two tables differ greatly, but for the most part, the document ranking has not changed much for the two tables. The document ranking changed the most with the query "car". For example, in the first table, the second document is tied with being the least relevant document; however, in the second table, it is the second most relevant document. This particular example shows that terms not in the query can have different distribution in relevant documents than in documents overall. Also, notice that the word "car" occurs in most of the queries. Because the query "car" is so generic, almost all the documents were relevant to some extent. In the first table, it is impossible to distinguish between documents with just one common word. In the second table, there is more variation among the documents' relevance scores for the "car" column. This variation is due to the fact that we are using features that were not present in the query. This implies that sometimes the words in the query itself are not enough information to decide if a document is relevant or not.

4. We are no longer computing the probability that the document is relevant. Instead, we are computing a quantity that is equivalent to this probability under rank. As a result of some of the simplifications that we made (e.g. dropping the k'/k term) it is possible to get a relevance score that is greater than 1.

5. This assumption seems reasonable for many real world situations (in fact, it is built into the VSM). When users search for documents, they usually want documents that contain one of the terms they searched for, for example. You could implement a more efficient retrieval system by precomputing an index that tells you which documents have each feature. Then you could use this index to obtain all of the documents that have at least one feature in common with the query and estimate the relevance of the documents in this subset.

6 References

1. Stephen Robertson and Karen Spärck Jones. Relevance weighting of search terms. Journal of the American Society for Information Science 27(3): 129-46 (1976). The probabilistic argument is presented in the appendix.

2. William S. Cooper. Some inconsistencies and misidentified modeling assumptions in probabilistic information retrieval. ACM Transactions on Information Systems (TOIS), pp. 100-111, 1995.

Code Listing 1: Python code for problem 1b

```
from collections import defaultdict

docs = ["this red car is fast and new toyota is a good brand of car",
        "this automobile is red and the brand is toyota it is not very fast but it is cheap",
        "this green car is fast and the mileage is low",
        "the car with low mileage is under a tree",
        "the green tree is in the park",
        "the automobile is green and this car is cheap",
        "park the car under the tree",
        "the toyota has low mileage and is very cheap",
        "this car is not a good brand",
        "there is a automobile that is red and in the park"]

# split each doc into words
docs = [d.split() for d in docs]

queries = ["car", "toyota", "park", "green car low mileage", "toyota brand car"]
queries = [q.split() for q in queries]

# (i, j) is in the list if query qi is relevant to doc j
query_doc_relevant = [(1,1), (2,1), (5,1), (1,2), (2,2), (5,2), (1,3), (4,3), (1,4),
                      (3,5), (1,6), (1,7), (3,7), (1,8), (2,8), (1,9), (1,10), (3,10)]
query_doc_relevant = [(x-1, y-1) for x, y in query_doc_relevant]

def word_probs(docs):
    """ word_probs(docs)[w] is the fraction of docs that contain w """
    probs = defaultdict(float)
    for d in docs:
        for w in set(d):
            probs[w] += 1.0 / len(docs)
    return probs

def make_table(only_terms_in_query):
    """ Returns d such that d[i][j] is the probability document i is
    relevant to query j using the RSJ model. If only_terms_in_query is
    True, then equation 16 is used, otherwise equation 17 is used."""
    # estimate Pr(F[j])
    word_priors = word_probs(docs)
    doc_query_prob = defaultdict(lambda: defaultdict(float))
    for qi, query in enumerate(queries):
        rel_docs = [docs[dj] for qj, dj in query_doc_relevant if qj == qi]
        word_given_rel = word_probs(rel_docs)
        # estimate probability doc di is relevant to query qi
        # this is the product of
        for di, doc in enumerate(docs):
            doc_query_prob[qi][di] = 1.0
            # only use the words in the query if only_terms_in_query is True
            for w in (query if only_terms_in_query else word_priors.keys()):
                if w in set(doc):
                    # Pr(F[j] = 0 | r, q)/Pr(F[j] = 0)
                    doc_query_prob[qi][di] *= word_given_rel[w] / word_priors[w]
                else:
                    # Pr(F[j] = 1 | r, q)/Pr(F[j] = 1)
                    doc_query_prob[qi][di] *= \
                        (1.0 - word_given_rel[w])/(1.0 - word_priors[w])
```

```
    return doc_query_prob

def print_table(doc_query_probs):
    for di, doc in enumerate(docs):
        row_probs = [doc_query_probs[qi][di] for qi in xrange(len(queries))]
        print ' '.join(("%.2f" % p).ljust(9) for p in row_probs)

print 'the entry at row i and col j is the probability doc i is relevant to query j'
print 'just terms in query'
print_table(make_table(True))
print 'all terms'
print_table(make_table(False))
```