Probabilistic Information Retrieval

The probabilistic approach to information retrieval provides a different formal basis for a retrieval model and results in different techniques for setting term weights.

Given only a query, an IR system has uncertain understanding of the information need. Given the query and document representations, a system has an uncertain guess of whether a document has content relevant to the information need. Probability theory provides a principled foundation for such reasoning under uncertainty.

Idea behind probabilistic IR is that we can try to estimate the probability of a document being relevant giving the the terms and the query.

We will use some assumption to simplify the computation of this probability.

The odds of an event A (which provide a kind of multiplier for how probabilities change) is defined as: $O(A) = \frac{P(A)}{P(\bar{A})} = \frac{P(A)}{1-P(A)}$, if we use some monotone function of the probability, the ranking will remain the same, so we are going to use odds.

Recall also the Bayes' rule: $P(A|B) = \frac{P(B|A)P(A)}{P(B)} = \frac{P(B|A)}{\sum_{X\in[A,\bar{A}]} P(B|X)P(X)} P(A)$, which means that given the prior $P(A)$ (we start off with an initial estimate of how likely the event A is when we do not have any other information) we can update it based on the evidence B, thus obtaining a posterior probability $P(A|B)$.

Probability Ranking Principle

For each document we consider an indicator random variable R_{dq} representing whether a document d is relevant (1) or not (0) w.r.t. a query q. Given a set of documents, we want to rank them according to the probability of the document being relevant, given the document and the query. Hence we want to compute $P(R = 1|d, q)$ (this is the basis of **probability ranking principle (PRP)**), we want to compute or approximate it in an efficient way.

We can consider what is the best way of taking this decision according to the **1/0 loss**: we have a penalty when we retrieve a document that is not relevant, we have a penalty when we miss a relevant document. The penalty is the same in all cases, there is no cost associated to retrieving documents (this is a binary situation in which we are evaluating our accuracy).

If we need to rank documents we use decreasing $P(R = 1|d, q)$ (this is the PRP). If we need to return a set of documents we return all the ones where $P(R = 1|d, q) > P(R = 0|d, q)$ (this is the decision that minimises the risk of loss). It can be proved that this choice minimise the *expected* loss under the 1/0 loss.

We can expand this idea and change which document to retrieve using a more complex model for costs (i.e. we assume a model for retrieval cost):

- C_1 is the cost of retrieving a relevant document;
- C_0 is the cost of retrieving a non-relevant document.

In order to select the documents to be retrieved d we pick the ones where for all non-retrieved documents d' it holds that:

$$C_1 \cdot P(R = 1|d, q) + C_0 \cdot P(R = 0|d, q) \leq C_1 \cdot P(R = 1|d', q) + C_0 \cdot P(R = 0|d', q)$$

i.e. the weighted cost of retrieving $d \leq$ weighted cost of retrieving d' (this is the PRP in this nwe model of costs).

Binary independence model

In binary independence model (BIM) assume:

- **binary/boolean assumption**: each document (or query) is represented as a vector $\bar{x} = (x_1, \dots, x_M)$ where $x_i = 1$ if the term is present and $x_i = 0$ otherwise (binary term incidence vectors).
- **independence**: we assume that all terms occurs in a document independently, the model recognises no association between terms (not correct, but works decently).
- we also assume that the relevance of a document independent from all the other documents in the collection (which might not be true, e.g. near duplicate documents). We also had this assumption also in the other two models we have seen (e.g. the similarity score was computed without considering which other documents were in the collection).

Estimation of the probability

To make a probabilistic retrieval strategy precise, we need to estimate how terms in documents contribute to relevance: specifically we wish to know how term frequency, document frequency, document length and other statistics that we can compute influence judgement about document relevance, and how they can be reasonably combined to estimate the probability of document relevance.

Both the query and the document are represented as vectors, hence $P(R = 1|d, q)$ is actually $P(R = 1|\bar{x}, \bar{q}) = \frac{P(\bar{x}|R=1,\bar{q})P(R=1|\bar{q})}{P(\bar{x}|\bar{q})}$, analogous for $R = 0$.

Note that $P(\bar{x}|R = 1, \bar{q})$ is the probability for a document with representation $\bar{x}$ is retrieved given that a relevant document for the query q is retrieved, $P(R = 1|\bar{q})$ is the (prior) probability of retrieving a relevant document for the query q.

We can repeat the same reasoning for $R = 0$ and get $P(R = 0|\bar{x}, \bar{q}) = \frac{P(\bar{x}|R=0,\bar{q})P(R=0|\bar{q})}{P(\bar{x}|\bar{q})}$, where $P(\bar{x}|R = 0, \bar{q})$ is the probability for a document with representation $\bar{x}$ is retrieved given that a non-relevant document for the query q is retrieved and $P(R=0|\bar{q})$ is the (prior) probability of retrieving a non-relevant document for the query q.

Because a document is either relevant or non-relevant to a query, we must have: $P(R=1|\bar{x},\bar{q}) + P(R=0|\bar{x},\bar{q}) = 1$.

We don't need the exact probability (since we are interested only in the ranking of documents), we can use a monotone function of the probability, such as the odds, as

$$O(R|\bar{x},\bar{q}) = \frac{P(R=1|\bar{x},\bar{q})}{P(R=0|\bar{x},\bar{q})} = \frac{P(\bar{x}|R=1,\bar{q})P(R=1|\bar{q})}{P(\bar{x}|R=0,\bar{q})P(R=0|\bar{q})}$$

We see that the first part of this formula depends on the document, the second part is the same for all documents, so it does not affect the ranking and we can remove it. Now we have to estimate $\frac{P(\bar{x}|R=1,\bar{q})}{P(\bar{x}|R=0,\bar{q})}$.

Now we can use our hypothesis, in particular the independence assumption (for which each of the terms is assumed to appear independently from the others), which means that the value to estimate is now: $\prod_{i=1}^{M} \frac{P(x_i|R=1,\bar{q})}{P(x_i|R=0,\bar{q})}$

Furthermore, we are walking in the binary model, so the term can be either present or not present, so we can split our formula in product of probabilities of terms in the documents and probability of terms that are not in the document.

Hence: $\prod_{i:x_i=1} \frac{P(x_i=1|R=1,\bar{q})}{P(x_i=1|R=0,\bar{q})} \cdot \prod_{i:x_i=0} \frac{P(x_i=0|R=1,\bar{q})}{P(x_i=0|R=0,\bar{q})}$

This probabilities appear for each document. We need to estimate the four probabilities of the following contingency table:

	Document relevant (R=1)	Document not relevant (R=0)
Term present ($x_i = 1$)	p_i	u_i
Term absent ($x_i = 0$)	$1 - p_i$	$1 - u_i$

We can replace as $\prod_{i:x_i=1} \frac{p_i}{u_i} \cdot \prod_{i:x_i=0} \frac{1-p_i}{1-u_i}$.

We can moreover assume that terms not occurring in the query appears equally in relevant and non-relevant documents, i.e. $p_i = u_i$ when $q_i = 0$, so we can remove the factors for all terms not in the query (hence considering only terms in the products that appear in the query), obtaining: $\prod_{i:x_i=1;q_i=1} \frac{p_i}{u_i} \cdot \prod_{i:x_i=0;q_i=1} \frac{1-p_i}{1-u_i}$.

The left product is over query terms found in the document, the right product is over query terms not found in the document.

We multiply everything by $\prod_{i:x_i=1;q_i=1}$ by $\prod_{i:x_i=1;q_i=1} \frac{1-p_i}{1-u_i} \cdot \frac{1-u_i}{1-p_i}$ (each term is

1), we get $\prod_{i:x_i=1;q_i=1} \frac{p_i}{u_i} \frac{1-u_i}{1-p_i} \prod_{i:q_i=1} \frac{1-p_i}{1-u_i}$.

So now we have to compute probabilities only for the terms that are both in the query and in the document. Moreover the second factor does not depend on the document and we can remove it.

Ratio of odds

If we look closer at the previous formula, we can see that we have a ratio of two odds, the odds of the term appearing in the document if the document is relevant, and the inverse odds of the term appearing in the document if the document is not relevant. We need to estimate these both. I.e. we need to compute the odds of the term appearing in the document if the document is relevant $\frac{p_i}{1-p_i}$ and the inverse odds of the term appearing in the document if the document is not relevant $\frac{1-u_i}{u_i}$.

At the end, for ranking we will use the **retrieval status value (RSV)** for a certain document d, which is defined as:

$$RSV_d = log\left(prod_{i:x_i=1;q_i=1} \frac{p_i}{u_i} \frac{1-u_i}{1-p_i}\right) = \sum_{i:x_i=q_i=1} log \frac{p_i(1-u_i)}{u_i(1-p_i)}$$

We call c_i each term of the sum and we can notice that c_i is the sum of two terms (p_i is the probability of getting the term inside the document when the document is relevant, u_i is the probability of getting the term inside the document, but the document is not-relevant): $c_i = log\frac{p_i}{1-p_i} + log\frac{1-u_i}{u_i}$. The first addend in the log-odds of the term appearing if the document is relevant, the second is the log-odds if the term appearing if the document is not relevant. It can be considered as the weight of the i-th term of the dictionary, and can be pre-computed.

At the end the RSV of a document d is the sum of the weights for all the terms contained both in the document and in the query: $RSV_d = \sum_{i:x_i=q_i=1} c_i$.

So we moved for computing the probability to summing weights, some of them can be precomputed.

Probability estimation in practice

So now, for each term, we need to compute both p_i and u_i.

We have two cases: relevant (estimate p_i) and non-relevant (estimate u_i) documents.

We assume that the non-relevant are the majority inside the collection, so we approximate the probability of the non-relevant documents with statistics computed using the entire collection (population-wide statistics).

We usually estimate $log\frac{1-u_i}{u_i} = log\frac{N-df_i}{df_i}$ for a term i. Actually we can approximate it by $log\frac{N}{df_i}$ (since $df_i << N$), which is actually the inverse document frequency of the term i.

We still have to estimate $log\frac{p_i}{1-p_i}$, i.e. the estimation for relevant document (p_i). To do this we have to know which are the relevant documents, to compute statistics on them (cannot use population-wide statistics). Another possibility is to put all the probability to 0.5 (the most brutal way of doing so, in this way the first term of c_i is 0), with this estimate and assuming idf_i for non-relevant documents, this approximation is the sum of the idf_i for all query terms that occurs in the document ($1 - p_i$ and p_i factors cancel out in the expression for RSV). Otherwise we can consider some collection-level statistics, for example $p_i = \frac{df_i}{N}$.

This is what we can do, unless we have some kind of feedback form the user, and we can combine it with relevance feedback. We start with some raw estimates of p_i and u_i, then we retrieve the results to the user and we use some kind of relevance feedback. We obtain $VR = \{d \in V : R_{d,q} = 1\}$ and we recompute our estimates for p_i and u_i, V is the subset of documents for which we have the feedback form the user (probabilities are not fixed in this context).

We assume that VR is large enough, otherwise our estimation is large enough, so we can know estimate the two probabilities: $p_i = \frac{|VR_i|}{|VR|}$ and $u_i = \frac{df_i - |VR_i|}{N - |VR|}$. If a term is not contained in any document, then $p_i = 0$, so we use a smoothed version $p_i = \frac{|VR_i| + \frac{1}{2}}{|VR|+1}$ and $u_i = \frac{df_i - |VR_i| + \frac{1}{2}}{N - |VR| + 1}$.

We can also extend the previous model to allow for pseudo-relevance feedback: select the first k highest ranked documents, consider them as a set V. Consider all of them relevant, and update the probability accordingly: $p_i = \frac{|V_i| + \frac{1}{2}}{|V|+1}$ and $u_i = \frac{df_i - |V_i| + \frac{1}{2}}{N - |V| + 1}$. Repeat until convergence.

Okapi BM25

Okapi BM25 is a variant of the binary independence model, we still have the independence part, but we drop the binary part. Indeed it takes into account the *frequency* of the terms inside the document.

We start with $RSV_d = \sum_{t \in q} idf_i$ and we weight it with a term that multiplies the idf which contains the $tf_{t,d}$, the length of the document L_d and the average length of the documents in the collection L_{avg}, so to have:

$$RSV_d = \sum_{t \in q} idf_t \cdot \frac{(k_1 + 1) tf_{t,d}}{k_1((1-b) + b \cdot \frac{L_d}{L_{abg}}) + tf_{t,d}}$$

k_1 is how important is the term frequency (positive tuning parameter that calibrates the document term frequency scaling, with $k_1 = 0$ we resort to binary model), its coefficient in the denominator is how much to normalise with respect to length (it determines the scaling of the document length), regulated by b ($b = 0$ no normalisation, $b = 1$ full scaling by document length).

In general, k_1 and b are two parameters, with $b \in [0, 1]$ and $k_1 \geq 0$, usually $k_1 \in [1.2, 2.0]$.

Bayesian Networks

We now want to add dependency among the terms, we are removing both assumption of the binary independent model.

Bayesian networks are a kind of *graphical model* using a directed acyclic graph to show probabilistic dependencies among variables. They are useful because when we need to compute $P(y|x_i, \dots, x_k)$ we can reduce to compute $P(y|Pa(y))$.

In order to find the probabilities of an event we can use the tables of conditional probabilities of the network (more than binary variables makes larger tables). The size of the table depends on the number of edges entering the node. For binary variables it is 2^k with k the in-degree of the node.

Inference in Bayesian networks is, in general case, intractable form a computational point of view, but for specific cases it can still be performed efficiently.

Bayesian Networks in IR

BN can model dependencies between terms or documents (contrarily to the assumption of the BIM). However, we must always keep an eye to complexity.

The simple structure is made by nodes representing the terms and nodes representing the documents, we have an edge when term is contained in the document.

If we don't know if a term is relevant or not, we say that the term is relevant with probability $\frac{1}{M}$ and non-relevant with probability $1 - \frac{1}{M}$.

For the documents we have different different levels of relevance, the size of the table of conditionally probability depends exponentially on the number of terms in the document.

So we need a different approach to store conditional probabilities. We assign a weight to each edge, the probability of a document be relevant given what we know about its parents is $P(d_j|Pa(d_j)) = \sum_{i:t_i \in Pa(d_j), t_i = 1} w_{i,j}$, i.e. sum all $w_{i,j}$ for all the parent nodes with state 1 (relevant).

Still we have a problem because we have to set the weights, at the end we need to have a conditional probability, so we want the weights to be positive and sum to at most one.

One possible way is to use the $tf-idf$ of tern i in document j squared, and normalise across all documents:

$$w_{i,j} = \alpha^{-1} \frac{tf - idf_{i,j}^2}{\sum_{t_k \in d_j} (tf - idf_{k,j})^2}$$

In order to answer a query, we assume that when we get the query, as a set of terms, all the terms in the query are relevant. So $t_i = 1$ for all terms in the query. So we have: $P(d_j|q) = \sum_{i:p_i \in Pa(d_j)} w_{i,j} P(t_j|q)$.

What we may want to add is some kind of dependency between different terms, while keeping the graph acyclic. What changes is that we now have to set the probabilities for words that can have dependencies with other terms. For root nodes (no parents) we use the same probabilities as before. We can use the idea of the Jaccard coefficient of similarity among terms. Given a configuration x of parent terms (which terms are present and which are not), let $A_{i,x}$ be the set of documents not containing t_i and containing the exact configuration x of the parent node. Similarly define $A_{t,}$ and A_x, then:

$$P(t_i = 0|Pa(t_i) = x) = \frac{|A_{t,x}|}{|A_{t_i}| + |A_x| - |A_{t_i,x}|}$$

$$P(t_i = 1|Pa(t_i) = x) = 1 - P(t_i = 0|Pa(t_i) = x)$$

We can also in principle add dependencies between documents. But as with terms, also for documents, we still have the problem that we have some conditional probability to estimate (must find a way to design or learn the dependencies, which topology of the network to use).