Information retrieval

Probabilistic IR

Corso di Laurea Magistrale in Informatica

Università di Roma Tor Vergata

Prof. Giorgio Gambosi

a.a. 2021-2022

Derived from slides originally produced by C. Manning and by H. Schütze

Probabilistic Approach to Retrieval

- Given a user information need (represented as a query) and a collection of documents (transformed into document representations), a system must determine how well the documents satisfy the query
  - An IR system has an uncertain understanding of the user query, and makes an uncertain guess of whether a document satisfies the query
- Probability theory provides a principled foundation for such reasoning under uncertainty
  - Probabilistic models exploit this foundation to estimate how likely it is that a document is relevant to a query

Why probabilistic?

At first glance, a document d is either relevant or not relevant wrt to a query q

Indeed there are several sources of uncertainty:

- wrt different users: different users may have different opinions regarding the relevance of d wrt q
- wrt the same user in different contexts: a user may judge d relevant or not in dependence of many factors, in different contexts
- wrt the document representation: d is usually represented in some way in the IR system; hence relevance is estimated on limited information
- wrt the IR system: the system itself may induce approximations/errors in estimating the relevance of d

Probabilistic IR Models at a Glance

In a probabilistic model, document are retrieved/ranked by their (estimated) probability of being relevant, given the query

$$p(d \text{ is relevant} | q)$$

Probabilistic IR Models at a Glance

- Classical probabilistic retrieval model
  - Probability ranking principle
    - Binary Independence Model, BestMatch25 (Okapi)
- Language model approach to IR
- ...

Probabilistic methods are one of the oldest but also one of the hottest topics in IR

Probabilistic vs. vector space model

- Vector space model: rank documents according to similarity to query.
- The notion of similarity does not translate directly into an assessment of "is the document a good document to give to the user or not?"
  - The most similar document can be highly relevant or completely nonrelevant.
- Probability theory is arguably a cleaner formalization of what we really want an IR system to do: give relevant documents to the user.

- For events A and B
  - Joint probability $p(A \cap B)$ of both events occurring
  - Conditional probability $p(A|B)$ of event A occurring given that event B has occurred
- Chain rule gives fundamental relationship between joint and conditional probabilities:

$$p(A, B) = p(A \cap B) = p(A|B) \cdot p(B) = p(B|A) \cdot p(A)$$

- Similarly for the complement of an event $p(\overline{A})$:

$$p(\overline{A}B) = p(B|\overline{A}) \cdot p(\overline{A})$$

- Partition rule: if B can be divided into an exhaustive set of disjoint subcases, then $p(B)$ is the sum of the probabilities of the subcases. A special case of this rule gives:

$$p(B) = p(A, B) + p(\overline{A}, B)$$

Bayes' Rule for inverting conditional probabilities:

$$p(A|B) = \frac{p(B|A) \cdot p(A)}{p(B)} = \left[\frac{p(B|A)}{\sum_{X \in \{A, \overline{A}\}} p(B|X) \cdot p(X)} \right] p(A)$$

Can be thought of as a way of updating probabilities:

- Start off with prior probability $p(A)$ (initial estimate of how likely event A is in the absence of any other information)
- Derive a posterior probability $p(A|B)$ after having seen the evidence B, based on the likelihood of B occurring in the two cases that A does or does not hold

Odds of an event provide a kind of multiplier for how probabilities change:

$$\text{Odds:} \qquad O(A) = \frac{p(A)}{p(\overline{A})} = \frac{p(A)}{1 - p(A)}$$

Probabilistic relevance

- Assume binary notion of relevance: $R_{d,q}$ is a random binary variable, such that
  - $R_{d,q} = 1$ if document d is relevant w.r.t query q
  - $R_{d,q} = 0$ otherwise
- We may interpretate $p(R_{d,q}) = p(R|d,q)$ as the probability that a random user judges d relevant for query q. In other words, assuming an event space defined on a set U of users (or user types), it is

$$p(R|d,q) = \sum_{u \in U} p(R|d,q,u)p(u)$$

where $p(R|d,q,u)$ is the probability that user u judges d relevant for query q, and $p(u)$ is the probability that u is the user asked to judge relevance

Probabilistic relevance

- documents could be retrieved by applying the Bayes decision rule, that is if $p(R|d,q) > p(\overline{R}|d,q)$ hence if, using odds,

$$O(R|d,q) = \frac{p(R|d,q)}{p(\overline{R}|d,q)} > 1$$

- we assume no relevance judgement from users is available

Probabilistic relevance

$p(R|d,q)$ (and $p(\overline{R}|d,q)$ can be decomposed in two ways

- $$p(R|d,q) = \frac{p(d|R,q)p(R|q)}{p(d|q)} \qquad p(\overline{R}|d,q) = \frac{p(d|\overline{R},q)p(\overline{R}|q)}{p(d|q)}$$

  that is, we look at the probability of relevant and not relevant documents when the query is fixed. This is the approach of BIM, 2-Poisson and BM25 models

- $$p(R|d,q) = \frac{p(q|R,d)p(R|d)}{p(q|d)} \qquad p(\overline{R}|d,q) = \frac{p(q|\overline{R},d)p(\overline{R}|d)}{p(q|d)}$$

  that is, we consider the probability of the query wrt to relevant and not relevant documents. This is the approach of language models

Probabilistic Ranking

- Ranked retrieval setup: given a collection of documents, the user issues a query, and an ordered list of documents is returned
- Probabilistic ranking orders documents decreasingly by their estimated probability of relevance w.r.t. query: $p(R_{d,q}) = p(R|d, q)$
- in order to estimate and compare $p(R|d, q)$ and $p(R|d', q)$ several simplifying assumptions are done
  - Independence assumption: the relevance of each document is independent of the relevance of other documents

Probabilistic Ranking

Let us consider the approach of considering

$$p(R|d,q) = \frac{p(d|R,q)p(R|q)}{p(d|q)} = \frac{p(d|R,q)p(R|q)}{p(d)}$$

where:

- $p(d|R,q)$ is the probability that document d is randomly sampled from the subcollection of documents relevant for query q
- $p(R|q)$ is the probability that a random document from the collection is relevant for q
- $p(d|q) = p(d)$ (we assume d and q independent) is the probability that document d is sampled from the collection.

The same clearly holds for non relevant documents

$$p(\overline{R}|d,q) = \frac{p(d|\overline{R},q)p(\overline{R}|q)}{p(d|q)} = \frac{p(d|\overline{R},q)p(\overline{R}|q)}{p(d)}$$

moreover, either a document is relevant or it is non relevant, that is $p(\overline{R}|d,q) + p(R|d,q) = 1$

- Assumptions:
  - uniform document probability: $p(d) = p(d')$ for all d, d' (this could not be true if we consider document representations, but assume it holds, for the sake of simplicity)
  - $p(R|q)$ can be ignored: if we are interested in ranking documents, the probabilities $p(R|q)$ and $p(\overline{R}|q)$ are constant on all documents, and can be ignored

Probability Ranking Principle (Robertson, 1977)

- PRP in brief
  - If the retrieved documents (w.r.t a query) are ranked decreasingly on their probability of relevance, then the effectiveness of the system will be the best that is obtainable
- PRP in full
  - If [the IR] system's response to each [query] is a ranking of the documents [...] in order of decreasing probability of relevance to the [query], where the probabilities are estimated as accurately as possible on the basis of whatever data have been made available to the system for this purpose, the overall effectiveness of the system to its user will be the best that is obtainable on the basis of those data

Error cost of retrieval

- The overall goal of the IR system is to return the best possible results, in terms of relevance, as the top k documents, for any value of k the user chooses to examine. How to formalize "best"?
- We may associate an error cost for each document wrongly returned (if not relevant) or not returned (if relevant)
- The best result is then the one which minimizes the overall error cost
- In a probabilistic framework the relevance of a document wrt a query q is a random variable with an associated probability $p(R|d,q)$: the error cost can only be defined in terms of expectation

Error cost of retrieval

- Assume that the following costs are defined:
  - $C(d,q)$: the cost when d is a relevant document and it is not returned
  - $C'(d,q)$: the cost when d is a non relevant document and it is returned
- If a set $D(q)$ of documents is returned by the system, the expected cost (risk) is given by

$$
\begin{aligned}
R(D(q)) &= \sum_{d\in D(q)} C'(d,q)p(\overline{R}|d,q) + \sum_{d\notin D(q)} C(d,q)p(R|d,q) \\
&= \sum_{d\in D(q)} C'(d,q)(1-p(R|d,q)) + \sum_{d\notin D(q)} C(d,q)p(R|d,q)
\end{aligned}
$$

- We may use the risk associated to a specific result as an inverse measure of the quality of the result

Assume a simple binary error cost function is used, where $C(d,q) = C'(d,q) = 1$: that is, any case where a relevant document is not returned or a non relevant document is returned has a constant cost (say 1)

Then

$$R(D(q)) = \sum_{d \in D(q)} (1 - p(R|d,q)) + \sum_{d \notin D(q)} p(R|d,q) = |D(q)| + \sum_{d \notin D(q)} p(R|d,q) - \sum_{d \in D(q)} p(R|d,q)$$

Probability Ranking Principle

If we assume a fixed $|D(q)| = k$, that is the user is interested to the best k documents, then $R(D(q))$ is minimized if the set $D(q)$ is such that

$$\sum_{d \in D(q)} p(R|d,q) - \sum_{d \notin D(q)} p(R|d,q)$$

is maximized: this corresponds to $D(q)$ containing the k documents with highest probability of relevance $p(R|d,q)$.

Probability Ranking Principle

As a consequence of the above observations, the best retrieval policy is the one that, for any k, returns the k topmost documents in a ranking by non-increasing probability of relevance.

Theorem

When 0/1 loss is assumed, the PRP is optimal, in that it minimizes the (expected) loss

This clearly holds if all probabilities are correct

How do we compute all those probabilities?

- We do not know the exact probabilities, need of estimates
    - Binary Independence Model (BIM) is the simplest approach
- Assumptions:
    - Relevance of each document is independent of relevance of other documents (Risk of returning lot of duplicates)
    - Boolean model of relevance

Documents as set of features

Documents are represented for retrieval and ranking with regards to a specified set of features, that is a representation.

- d is represented as a vector $(f_1, \dots, f_n)$ of feature values
- this turns out to considering

$$p(R|f_1, \dots, f_n, q) = \frac{p(f_1, \dots, f_n|R, q)p(R|q)}{p(f_1, \dots, f_n)} \stackrel{\text{rank}}{=} p(f_1, \dots, f_n|R, q)$$

- Assumption:
  - feature (conditional) independence $p(f_1, \dots, f_n|R, q) = \prod_i p(f_i|R, q)$: this is the naive assumption of Naive Bayes models

Binary Independence Model (BIM)

In BIM, each feature:

1. is associated to a term
2. is binary: 1 if the term occurs, 0 if it does not occur
3. document d represented by vector $v_d = (x_1, \dots, x_m)$, where $x_i = 1$ iff term t_i occurs in d
4. query q represented by vector $v_q = (y_1, \dots, y_m)$, where $y_i = 1$ iff term t_i occurs in q
5. different documents/queries may have the same vector representation

The feature conditional assumption turn out to be a no association between terms assumption, conditioned on the query and the document relevance with respect to the query itself.

Binary incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
Anthony	1	1	0	0	0	1	
Brutus	1	1	0	1	0	0	
Caesar	1	1	0	1	1	1	
Calpurnia	0	1	0	0	0	0	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	0	
...							

Each document is represented as a binary vector $\in \{0, 1\}^{|V|}$.

Binary Independence Model

To make a probabilistic retrieval strategy precise, need to estimate how terms in documents contribute to relevance

- Find measurable statistics (term frequency, document frequency, document length) that affect judgments about document relevance
- Combine these statistics to estimate the probability $p(R|d, q)$ of document relevance

Binary Independence Model

$p(R|d,q)$ is modeled using term incidence vectors as $p(R|v_d, v_q)$

$$
\begin{aligned}
p(R|v_d, v_q) &= \frac{p(v_d|R, v_q)p(R|v_q)}{p(v_d|v_q)} \overset{\text{rank}}{=} p(v_d|R, v_q) \\
p(\overline{R}|v_d, v_q) &= \frac{p(v_d|\overline{R}, v_q)p(\overline{R}|v_q)}{p(v_d|v_q)} \overset{\text{rank}}{=} p(v_d|\overline{R}, v_q)
\end{aligned} \tag{1}
$$

- $p(v_d|R, v_q)$ and $p(v_d|\overline{R}, v_q)$: probability that if a relevant or nonrelevant document is retrieved, then that document's representation is v_d
- Use statistics about the document collection to estimate these probabilities

Deriving a Ranking Function for Query Terms (1)

- Given a query q, ranking documents by $p(R|d, q)$ is modeled under BIM as ranking them by $p(R|v_d, v_q)$
- Easier: rank documents by their odds of relevance (gives same ranking)

$$O(R|v_d, v_q) = \frac{p(R|v_d, v_q)}{p(\overline{R}|v_d, v_q)} \stackrel{\text{rank}}{=} \frac{p(v_d|R, v_q)}{p(v_d|\overline{R}, v_q)}$$

Deriving a Ranking Function for Query Terms (2)

By the Naive Bayes conditional independence assumption stated above, the presence or absence of a word in a document is independent of the presence or absence of any other word (given the query and the relevance of the document wrt the query):

$$\frac{p(v_d|R, v_q)}{p(v_d|\overline{R}, v_q)} = \prod_{i=1}^{M} \frac{p(x_i|R, v_q)}{p(x_i|\overline{R}, v_q)}$$

So:

$$O(R|v_d, v_q) \stackrel{\text{rank}}{=} \prod_{i=1}^{M} \frac{p(x_i|R, v_q)}{p(x_i|\overline{R}, v_q)}$$

Deriving a Ranking Function for Query Terms (3)

Since each x_i is either 0 or 1, we can separate the terms (term split):

$$O(R|v_d, v_q) \stackrel{\text{rank}}{=} \prod_{t_i : x_i = 1} \frac{p(x_i = 1|R, v_q)}{p(x_i = 1|\overline{R}, v_q)} \cdot \prod_{t_i : x_i = 0} \frac{p(x_i = 0|R, v_q)}{p(x_i = 0|\overline{R}, v_q)}$$

Deriving a Ranking Function for Query Terms

Additional simplifying assumption: terms not occurring in the query are equally likely to occur in relevant and nonrelevant documents (non query term assumption)

- If $y_i = 0$, then $p(x_i = 1|R, v_q) = p(x_i = 1|\overline{R}, v_q)$

Hence, we obtain

$$O(R|v_d, v_q) \overset{\text{rank}}{=} \prod_{t_i : x_i = y_i = 1} \frac{p(x_i = 1|R, v_q)}{p(x_i = 1|\overline{R}, v_q)} \cdot \prod_{t_i : x_i = 0, y_i = 1} \frac{p(x_i = 0|R, v_q)}{p(x_i = 0|\overline{R}, v_q)}$$

- The left product is over query terms found in the document and the right product is over query terms not found in the document

Deriving a Ranking Function for Query Terms

- Let $p_t = p(x_t = 1|R, v_q)$ be the probability of a term appearing in a document relevant for q
- Let $u_t = p(x_t = 1|\overline{R}, v_q)$ be the probability of a term appearing in a document nonrelevant for q
- This can be displayed as a contingency table:

	relevant document (R)	nonrelevant document ($\overline{R}$)
Term present ($x_t = 1$)	p_t	u_t
Term absent ($x_t = 0$)	$1 - p_t$	$1 - u_t$

All this results into

$$O(R|v_d, v_q) \overset{\text{rank}}{=} \prod_{t_i : x_i = y_i = 1} \frac{p_i}{u_i} \cdot \prod_{t_i : x_i = 0, y_i = 1} \frac{1 - p_i}{1 - u_i}$$

By including the query terms found in the document into the second product, but simultaneously dividing by them in the first product, it results:

$$O(R|v_d, v_q) \overset{\text{rank}}{=} \prod_{t_i : x_i = y_i = 1} \frac{p_i(1 - u_i)}{u_i(1 - p_i)} \cdot \prod_{t_i : y_i = 1} \frac{1 - p_i}{1 - u_i}$$

Deriving a Ranking Function for Query Terms

- The first product is still over query terms found in the document, but the right product is now over all query terms, hence constant for a particular query and can be ignored.
- The only value that needs to be estimated to rank documents w.r.t a query is the first productory

$$O(R|v_d, v_q) \stackrel{\text{rank}}{=} \prod_{t_i : x_i = y_i = 1} \frac{p_i(1-u_i)}{u_i(1-p_i)}$$

- We can equally rank documents by the logarithm of this term, since log is a monotonic function. This is named Retrieval Status Value (RSV):

$$RSV_d = \log \prod_{t :_i x_i = y_i = 1} \frac{p_i(1-u_i)}{u_i(1-p_i)} = \sum_{t_i : x_i = y_i = 1} \log \frac{p_i(1-u_i)}{u_i(1-p_i)}$$

Equivalent: rank documents using the log odds ratios for the terms t_i in the query:

$$c_i = \log \frac{p_i(1-u_i)}{u_i(1-p_i)} = \log \frac{p_i}{1-p_i} - \log \frac{u_i}{1-u_i}$$

- The odds ratio is the ratio of two odds:
  1. the odds $\frac{p_i}{1-p_i}$ of term t_i appearing in a document if the document is relevant
  2. the odds $\frac{u_i}{1-u_i}$ of term t_i appearing in a document if the document is nonrelevant

Term weight c_i in BIM

$$c_i = \log \frac{p_i}{1 - p_i} - \log \frac{u_i}{1 - u_i}$$

can be seen as a term weight.

- $c_i = 0$: term t_i has equal odds of appearing in relevant and nonrelevant documents
- $c_i > 0$: term t_i has higher odds to appear in relevant documents
- $c_i < 0$: term t_i has higher odds to appear in nonrelevant documents

Term weight c_i in BIM

Retrieval status value for document d:

$$RSV_d = \sum_{t_i : x_i = y_i = 1} c_i$$

- So BIM and vector space model are identical on an operational level, except that the term weights are different.
- In particular: we can use the same data structures (such as an inverted index) for the two models.

How to compute probability estimates

Which information can be used to compute the probabilities of a term t appearing in a relevant or non relevant document?

- $p_i = p(x_i = 1|R, v_q)$ probability that term x_i appears in a document relevant for q
- $u_i = p(x_i = 1|\overline{R}, v_q)$ probability that term x_i appears in a document nonrelevant for q

There are two possible scenarios:

- There are some documents which we consider relevant and/or not relevant
  - A training set of relevance judgements given by users is available
  - Relevance judgements may derive by a pseudo-relevance feedback method
- No information (relevance judgements) is available

How to compute probability estimates

First case: relevance judgement are available (fraction R is the number of relevant documents in the collection).

For each term t_i in a query, estimate c_i in the whole collection using a contingency table of counts of documents in the collection, where:

- N is the total numebr of documents
- R is the number of relevant documents as derived from relevance judgement
- r_i is the number of relevant documents in which term t_i occurs
- df_{t_i} is the number of documents that contain term t_i

How to compute probability estimates

	Relevant documents	Non relevant documents	**Total**
Term present $x_i = 1$	r_i	$\mathrm{df}_{t_i} - r_i$	df_{t_i}
Term absent $x_i = 0$	$R - r_i$	$(N - \mathrm{df}_{t_i}) - (R - r_i)$	$N - \mathrm{df}_{t_i}$
Total	R	$N - R$	N

The resulting probability estimates (by maximum likelihood) are then:

$$p_i = \frac{r_i}{R}$$

$$u_i = \frac{\mathrm{df}_{t_i} - r_i}{N - R}$$

$$c_i = \log \frac{\frac{r_i}{R - r_i}}{\frac{\mathrm{df}_{t_i} - r_i}{(N - \mathrm{df}_{t_i}) - (R - r_i)}}$$

Avoiding zeros: smoothing

- If any of the counts is zero, then the term weight is not well-defined.
- Maximum likelihood estimates do not work for rare events.
- To avoid zeros: add a constant α to each count, for example, $\alpha = 0.5$
- For example, use $R - r_i + 0.5$ in formula for $R - r_i$

Exercise

- Query: `Obama health plan`
- Doc1: `Obama rejects allegations about his own bad health`
- Doc2: `The plan is to visit Obama`
- Doc3: `Obama raises concerns with US health plan reforms`

Estimate the probability that the above documents are relevant to the query. Use a contingency table. These are the only three documents in the collection

Simplifying assumption

- Assumption: relevant documents are a very small percentage of the collection. Consequence: statistics for nonrelevant documents can be approximated by statistics from the whole collection
- Hence, the probability of term occurrence in nonrelevant documents for a query is $u_i \approx \frac{\mathrm{df}_{t_i}}{N}$ and

$$\log \frac{1-u_i}{u_i} = \log \frac{N - \mathrm{df}_{t_i}}{\mathrm{df}_{t_i}} \approx \log \frac{N}{\mathrm{df}_{t_i}}$$

- This results into

$$c_i = \log \frac{p_i(1-u_i)}{u_i(1-p_i)} \approx \log \frac{p_i}{(1-p_i)} + \log \frac{N}{\mathrm{df}_{t_i}}$$

Probability estimates in adhoc retrieval

No relevance judgement available (ad-hoc retrieval)

- Assume that p_i is constant over all terms x_i in the query and that $p_i = 0.5$
- Each term is equally likely to occur in a relevant document, and so the p_i and $(1 - p_i)$ factors cancel out in the expression for RSV

Probability estimates in adhoc retrieval

- Combining this method with the earlier approximation for u_i, the document ranking is determined simply by which query terms occur in documents scaled by their idf weighting

$$RSV_d = \sum_{t_i : x_i = y_i = 1} \log \frac{p_i(1-u_i)}{u_i(1-p_i)} \approx \sum_{t_i : x_i = y_i = 1} \log \frac{N}{\mathrm{df}_{t_i}}$$

- For short documents (titles or abstracts) in one-pass (no relevance feedback) retrieval situations, this estimate can be quite satisfactory

How different are vector space and BIM?

- They are not that different.
- In either case you build an information retrieval scheme in the exact same way.
- For probabilistic IR, at the end, you score queries not by cosine similarity and tf-idf in a vector space, but by a slightly different formula motivated by probability theory.

Open issue: how to add term frequency and length normalization to the probabilistic model.

Key limitations of BIM

- BIM, like much of original IR, was designed for titles or abstracts, and not for modern full text search
- We want to pay attention to term frequency and document lengths
- Want some model of how often terms occur in docs

Introducing term frequency

Let us first remind the definition of the Retrieval Status Value:

$$RSV_d = \log \prod_{t:_i x_i = y_i = 1} \frac{p(x_i = 1|R, v_q)p(x_i = 0|\overline{R}, v_q)}{p(x_i = 1|\overline{R}, v_q)p(x_i = 0|R, v_q)}$$

By still applying the simplifying assumption introduced for BIM, we approximate $p(x_i = 1|\overline{R}, v_q)$ by $p(x_i = 1)$ and $p(x_i = 0|\overline{R}, v_q)$ by $p(x_i = 0)$

$$RSV_d = \sum_{t_i : x_i = y_i = 1} \log \frac{p(x_i = 1|R, v_q)p(x_i = 0)}{p(x_i = 1)p(x_i = 0|R, v_q)}$$

Introducing term frequency

Assume that we are representing documents in terms of count matrix (number of term occurrences). Then, document d has a representation as a vector of integers $(d_{t_1}, \dots, d_{t_n})$.

The Retrieval Status Value can be defined as

$$RSV_d = \sum_{t_i : y_i = 1} \log \frac{p(d_{t_i} = n_i | R, v_q) p(d_{t_i} = 0)}{p(d_{t_i} = n_i) p(d_{t_i} = 0 | R, v_q)}$$

How to estimate these probabilities?

Introducing term frequency

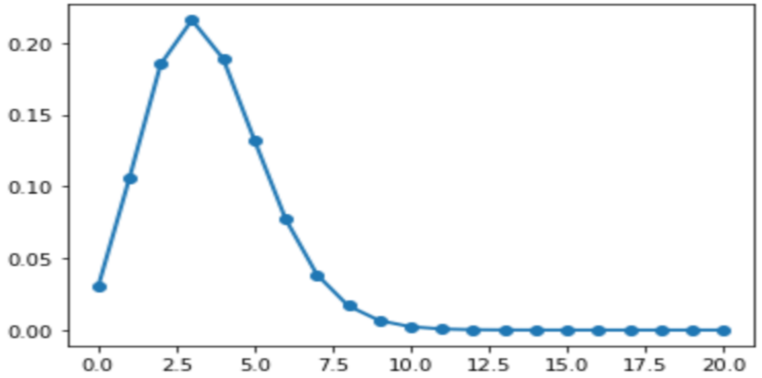
- We need an easy-to-compute discrete distribution to estimate p
- Simple choices:
  - Binomial distribution. Each document d has l word slots and each slot has a probability $\tilde{p}$ of having the term t_j , and $1 - \tilde{p}$ otherwise. The probability of a document having k occurrences of t_j is

    $$p(d_{t_j} = k) = \binom{k}{l} \tilde{p}^k (1 - \tilde{p})^{l-k}$$

    feasible scheme but the binomial coefficients can be messy
  - Poisson distribution. Assume (for now) all documents have same length l: term t_j occurs at some steady rate on average. Similar to a binomial for $l >> \tilde{p}$, but simpler to deal with: Binomial(l, $\tilde{p}$) modeled as Poisson($l\tilde{p}$)

General form of Poisson with mean λ:

$$\text{Poisson}(x|\lambda) = \frac{e^{-\lambda}\lambda^{x}}{x!}$$



Term occurrences as Poisson distribution

How to estimate λ?

The ratio between the collection frequency of t_j and the number of documents is often a good estimate

Term occurrences as Poisson distribution

This results in the following estimates:

- Let ρ_j the expected number of occurrences of t_j in documents relevant for q, then:

$$p(d_{t_j} = n_j | R, v_q) = \frac{e^{-\rho_j} \rho_j^{n_j}}{n_j!}$$
$$p(d_{t_j} = 0 | R, v_q) = e^{-\rho_j}$$

- Let γ_j the expected number of occurrences of t_j in documents in the collection, then:

$$p(d_{t_j} = n_i) = \frac{e^{-\gamma_j} \gamma_j^{n_j}}{n_j!}$$
$$p(d_{t_j} = 0) = e^{-\gamma_j}$$

Term occurrences as Poisson distribution

As a consequence:

$$RSV_d = \sum_{t_i : y_i = 1} \log \frac{p(d_{t_i} = n_i | R, v_q) p(d_{t_i} = 0)}{p(d_{t_i} = n_i) p(d_{t_i} = 0 | R, v_q)}$$

$$= \sum_{t_i : y_i = 1} \log \frac{\frac{e^{-\rho_j} \rho_j^{n_j}}{n_j!} e^{-\gamma_j}}{e^{-\rho_j} \frac{e^{-\gamma_j} \gamma_j^{n_j}}{n_j!}}$$

$$= \sum_{t_i : y_i = 1} \log \frac{\rho_j^{n_j}}{\gamma_j^{n_j}} = \sum_{t_j : y_j = 1} n_j \log \frac{\rho_j}{\gamma_j}$$

Each occurrence of t_j contributes to the score by a factor equal to the log of the ratio between its expected occurrences in relevant documents and its expected occurrences in general documents

Term occurrences as Poisson distribution

- The above assumptions fit rather well in the case of contentless terms, that is words which do not bear much meaning about a document topic
- In the case of contentful terms, which may characterize with their occurrence the topic of a document, the situation may be different

Term occurrences as Poisson distribution

Table 1. Frequency Distributions for 19 Word Types and Expected Frequencies Assuming a Poisson Distribution with $\lambda = 53/650$

	Word	Number of Documents Containing k Tokens													
Frequency	Type	k	0	1	2	3	4	5	6	7	8	9	10	11	12
51	act		608	35	5	2									
51	actions		617	27	2	0	2	0	2						
54	attitude		610	30	7	2	1								
52	based		600	48	2										
53	body		605	39	4	2									
52	castration		617	22	6	3	1	1							
55	cathexis		619	22	3	2	1	2	0	1					
51	comic		642	3	0	1	0	0	0	0	0	0	1	1	2
53	concerned		601	45	4										
53	conditions		604	39	7										
55	consists		602	41	7										
53	factor		609	32	7	1	1								
52	factors		611	27	11	1									
55	feeling		613	26	7	3	0	0	1						
52	find		602	45	2	1									
54	following		604	39	6	1									
51	force		603	43	4										
51	forces		609	33	6	2									
52	forgetting		629	11	3	2	2	1	1	0	0	0	1		
53	expected, assuming Poisson distribution		599	49	2										

Contentful words may have higher values of df_t: this happens for documents whose topic is described by the term

Two different Poisson distributions

Assume two types of terms occur in a document:

- Terms which do not characterize the topic of the document
- Terms which describe the topic of the document

Each class of terms is distributed according to a different Poisson: lower parameter for the first class, higher for the second class

Term occurrences as 2-Poisson distribution

The type of term in a document is modeled in terms of eliteness:

- What is eliteness?
  - Hidden binary variable for each document-term pair
  - Given a document, a term is elite if, in some sense, the document is about the concept denoted by the term: this implies that such term will tend to appear more often in the document
  - Term occurrences depend only on eliteness (not on relevance, at least directly)
  - But eliteness is related to relevance

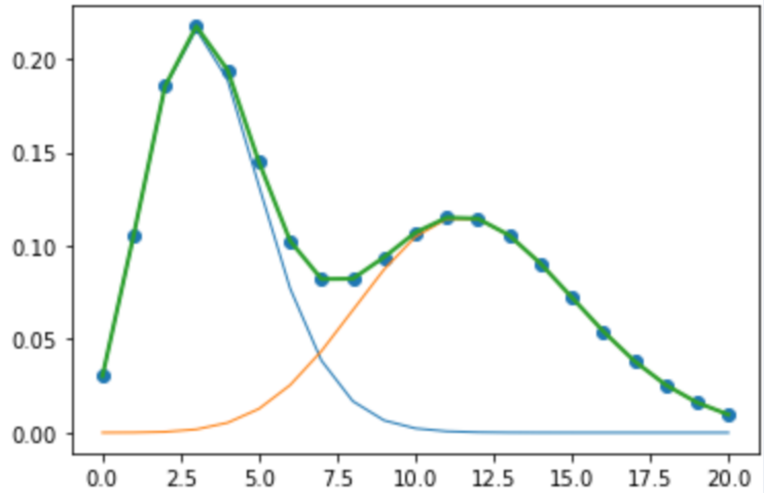
Term occurrences as 2-Poisson distribution

Let E_i denote the elite random variable for term t_i in the document considered. We assume that the distribution $p(d_{t_i} = n_i | R, v_q)$ can be expressed as the mixture of two Poisson distributions, for the elite and the not elite case.

$$p(d_{t_i} = n_i | R, v_q) = p(d_{t_i} = n_i | E_i) p(E_i | R, v_q) + p(d_{t_i} = n_i | \overline{E}_i) p(\overline{E}_i | R, v_q)$$
$$p(d_{t_i} = n_i | R, v_q) = p_i \cdot \text{Poisson}(n_i | \mu_i) + (1 - p_i) \cdot \text{Poisson}(n_i | \overline{\mu}_i)$$
$$p(d_{t_i} = n_i | R, v_q) = p_i \frac{e^{-\mu_i} \mu_i^{n_i}}{n_i!} + (1 - p_i) \frac{e^{-\overline{\mu}_i} \overline{\mu}_i^{n_i}}{n_i!}$$

where $p_i = p(E_i | R, v_q)$ is the probability that the document is elite for the term t_i

Term occurrences as 2-Poisson distribution



Term occurrences as 2-Poisson distribution

The probabilities in RSV can be decomposed as

$$\begin{aligned} p(d_{t_i} = n_i | R, v_q) &= C(n_i) p_i + \overline{C}(n_i)(1 - p_i) \\ p(d_{t_i} = 0 | R, v_q) &= C(0) p_i + \overline{C}(0)(1 - p_i) \\ p(d_{t_i} = n_i) &= C(n_i) \overline{p} + \overline{C}(n_i)(1 - \overline{p}) \\ p(d_{t_i} = 0) &= C(0) \overline{p} + \overline{C}(0)(1 - \overline{p}) \end{aligned}$$

where:

- $C(n_i) = \text{Poisson}(n_i|\mu_i)$ is the probability of observing n_i occurrences of the term if the document is elite for it
- $\overline{C}(n_i) = \text{Poisson}(n_i|\overline{\mu}_i)$ is the probability of observing n_i occurrences of the term if the document is not elite for it
- $p_i = p(E_i|R, v_q)$ is the probability that the document is elite for t_i assuming it is relevant
- $\overline{p} = p(E_i)$ is the probability that the document is elite for t_i assuming it is not relevant, estimated by considering the whole collection

Term occurrences as 2-Poisson distribution

The resulting RSV is then

$$RSV_d = \sum_{t_i : y_i = 1} \log \frac{(C(n_i)p_i + \overline{C}(n_i)(1-p_i))(C(0)\overline{p} + \overline{C}(0)(1-\overline{p}))}{(C(0)p_i + \overline{C}(0)(1-p_i))(C(n_i)\overline{p} + \overline{C}(n_i)(1-\overline{p}))}$$

The estimation of this expression requires, for each term t_i, the estimation of:

- the expectation μ_i, the average number of occurrences in an elite document
- the expectation $\overline{\mu}_i$, the average number of occurrences in a nonelite document
- the probability $p_i = p(E_i | R, v_q)$ that a document relevant for the query is elite for t_i
- the probability $\overline{p} = p(E_i)$ that any document in the collection is elite for t_i

This is way too difficult and expensive

Term occurrences as 2-Poisson distribution

We try to understand how the contribution

$$\log \frac{(C(n_i)p_i + \overline{C}(n_i)(1-p_i))(C(0)\overline{p} + \overline{C}(0)(1-\overline{p}))}{(C(0)p_i + \overline{C}(0)(1-p_i))(C(n_i)\overline{p} + \overline{C}(n_i)(1-\overline{p}))}$$

of a term t_i behaves in terms of n_i, observing that:

- for $n_i = 0$ it is 0
- for $n_i \to \infty$ it asymptotically approaches the value for the binary case

$$\log \frac{p_i(1-\overline{p})}{(1-p_i)\overline{p}}$$

- moreover, the function monotonically increases for $n_i > 0$
- we approximate the function with a simple parametric curve with the same qualitative properties

$$\frac{(k+1)n_i}{k+n_i} \log \frac{p_i(1-\overline{p})}{(1-p_i)\overline{p}}$$

Term occurrences as 2-Poisson distribution

In the case of no relevance/elite information available, we assume:

- $p(E_i|R, v_q) = 0.5$
- $p(E_i|\overline{R}, v_q) \approx p(E_i)$ can be further approximated by assuming $E_i = 1$ for all documents in wich t_i occurs

this results into

$$\log \frac{p_i(1-\overline{p})}{(1-p_i)\overline{p}} \approx \log \frac{N}{df_{t_i}}$$

and to a scoring function

$$RSV_d = \sum_{t_i : y_i = 1} \frac{(k+1)n_i}{k+n_i} \log \frac{N}{df_{t_i}}$$

This is a first step towards the BM25 model

Okapi BM25: Overview

- Okapi BM25 is a probabilistic model that incorporates term frequency (i.e., it's nonbinary) and length normalization.
- BIM was originally designed for short catalog records of fairly consistent length, and it works reasonably in these contexts
- For modern full-text search collections, a model should pay attention to term frequency and document length
- BestMatch25 (a.k.a BM25 or Okapi) is sensitive to these quantities
- BM25 is one of the most widely used and robust retrieval models

Okapi BM25: Starting point

- The simplest score for document d is just idf weighting of the query terms present in the document:

$$RSV_d = \sum_{t \in q} \log \frac{N}{\mathrm{df}_t}$$

Okapi BM25 first basic weighting

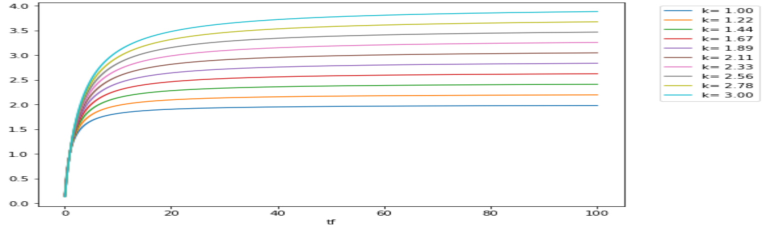
- Improve idf term $\log \frac{N}{\mathrm{df}_t}$ by factoring in term frequency.

$$RSV_d = \sum_{t \in q} \frac{(k_1 + 1)\mathrm{tf}_{td}}{k_1 + \mathrm{tf}_{td}} \log \frac{N}{\mathrm{df}_t}$$

- k_1: tuning parameter controlling the document term frequency scaling
- $(k_1 + 1)$ factor does not change ranking, but makes term score 1 when $\mathrm{tf}_{td} = 1$
- Similar to tf-idf, but term scores are bounded

Role of parameter k_1

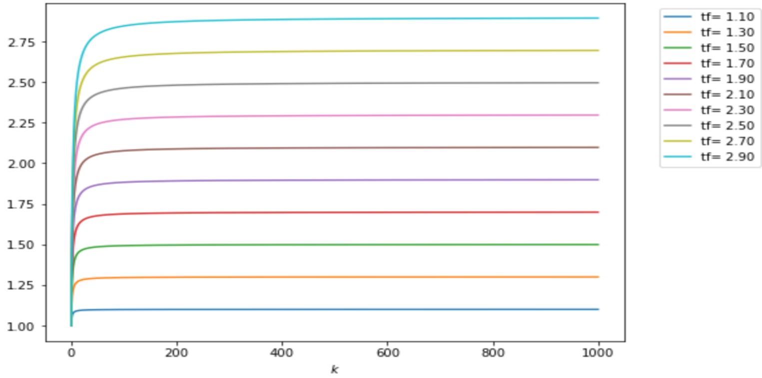
- k_1 helps determine term frequency saturation characteristics
- it limits how much a single query term can affect the score of a given document. It does this through approaching an asymptote



- A higher/lower k_1 value means that the slope of tf of BM25 curve changes. This has the effect of changing how terms occurring extra times add extra score.
- Usually, values around $1.2 - 2$

Exercise

- Interpret weighting formula for $k_1 = 0$
- Interpret weighting formula for $k_1 = 1$
- Interpret weighting formula for $k_1 \mapsto \infty$



Document length normalization

- Longer documents are likely to have larger tf_{td} values
- Why might documents be longer?
  - Verbosity: suggests observed tf_{td} too high
  - Larger scope: suggests observed tf_{td} may be right
- A real document collection probably has both effects so we should apply some kind of partial normalization

Document length normalization

- Document length

$$L_d = \sum_t \mathrm{tf}_{td}$$

- Document length average in the collection D

$$L_{\text{ave}} = \frac{1}{|D|} \sum_{d \in D} L_d$$

- Length normalization component

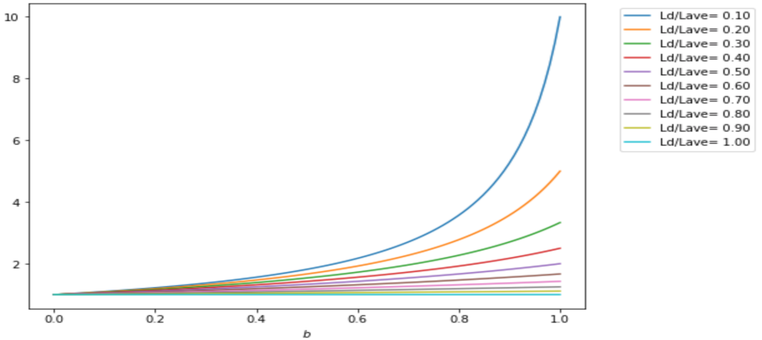
$$B = (1 - b) + b \frac{L_d}{L_{\text{ave}}} \qquad 0 \leq b \leq 1$$

  - $b = 1$: full document length normalization
  - $b = 0$: no document length normalization

Role of parameter b

B shows up in the denominator of RSV_d: longer documents correspond to higher L_d/L_{ave} and smaller RSV_d

- higher b results in smaller B (for a fixed L_d/L_{ave}) and higher RSV_d
- smaller b results in higher B (for a fixed L_d/L_{ave}) and smaller RSV_d
- Usually, b has a value around 0.75.



Okapi BM25 basic weighting

- Improve idf term $\log \frac{N}{\mathrm{df}_t}$ by factoring in term frequency and document length.

$$RSV_d = \sum_{t \in q} \frac{(k_1 + 1)\mathrm{tf}_{td}}{k_1((1 - b) + b\frac{L_d}{L_{\mathrm{ave}}}) + \mathrm{tf}_{td}} \log \frac{N}{\mathrm{df}_t}$$

- tf_{td}: term frequency in document d
- L_d (L_{ave}): length of document d (average document length in the whole collection)
- k_1: tuning parameter controlling the document term frequency scaling ($k_1 = 0$ is binary model, k_1 large is raw term frequency); usually around 1.2-2
- b: tuning parameter controlling the scaling by document length ($b = 0$ is no normalization, $b = 1$ is full normalization); usually around .75

Exercise

- Interpret BM25 weighting formula for $k_1 = 0$
- Interpret BM25 weighting formula for $k_1 = 1$ and $b = 0$
- Interpret BM25 weighting formula for $k_1 \mapsto \infty$ and $b = 0$
- Interpret BM25 weighting formula for $k_1 \mapsto \infty$ and $b = 1$

BM25 vs tf-idf

- Suppose your query is [machine learning]
- Suppose you have 2 documents with term counts:
  - doc1: learning 1024; machine 1
  - doc2: learning 16; machine 8
- Suppose that machine occurs in 1 out of 7 documents in the collection
- Suppose that learning occurs in 1 out of 10 documents in the collection
- tf-idf: $1 + \log_{10}(1 + tf) \log_{10}(N/df)$
  - doc1: 41.1
  - doc2: 35.8
- BM25: $k_1 = 2$
  - doc1: 31
  - doc2: 42.6

Okapi BM25 weighting for long queries

- For long queries, use similar weighting for query terms

$$RSV_d = \sum_{t \in q} \left[\log \frac{N}{\mathrm{df}_t} \right] \cdot \frac{(k_1 + 1)\mathrm{tf}_{td}}{k_1((1-b) + b \times (L_d / L_{\mathrm{ave}})) + \mathrm{tf}_{td}} \cdot \frac{(k_3 + 1)\mathrm{tf}_{tq}}{k_3 + \mathrm{tf}_{tq}}$$

- tf_{tq}: term frequency in the query q
- k_3: tuning parameter controlling term frequency scaling of the query
- No length normalization of queries (because retrieval is being done with respect to a single fixed query)
- The above tuning parameters should ideally be set to optimize performance on a development test collection. In the absence of such optimization, experiments have shown reasonable values are to set k_1 and k_3 to a value between 1.2 and 2 and $b = 0.75$

Which ranking model should I use?

- I want something basic and simple → use vector space with tf-idf weighting.
- I want to use a state-of-the-art ranking model with excellent performance → use BM25 (or language models) with tuned parameters
- In between: BM25 or language models with no or just one tuned parameter