Information retrieval

Paolo Boldi
DSI
LAW (Laboratory for Web Algorithmics)
Università degli Studi di Milan

Information retrieval basics

Information Retrieval (IR) aims at solving the following general kind of problems:

- Fix a *document collection* $\mathcal{D}$
- Fix a *query language* that allows you to build the set of queries $\mathcal{Q}$
- Given a query $q \in \mathcal{Q}$ (that supposedly corresponds to some information need by the user), find the documents $d \in \mathcal{D}$ that are *relevant* (i.e., pertinent, appropriate etc.) to q
- Probably also: sort them in decreasing order of appropriateness (*ranking*)

What is a document

In traditional IR, documents are just *text files*.

- The idea is that IR has to do with poorly structured data: documents have little or no structure, no semantics attached etc.
- Queries may have some structure, but in most cases they are similarly poorly structured.

These two facts mark the line separating IR from DB-retrieval and such.

Multi-field document

In some settings, documents may have more than one field. E.g., e-mails have a Subject, a Body, a Recipient etc.

- This is an embryonic form of structure.
- In cases like this one may consider it as if we had many *parallel collections*, and each is treated (indexed) separately.
- Then at query level one may mix collections with appropriate constructs; e.g., subject:urgent text:(meeting tomorrow)
- The same can be do implicitly (without the user required to be aware of it), when the query is solved and/or documents are ranked.

Other semi-structured documents

Sometimes, documents may have more structure.

- XML documents or alike. (*XML Information Retrieval*)
- Documents with a semantic description attached to them. (like in *Semantic web*).
- These situations can be dealt with enriching the query language and/or exploiting the richer structure when the query is solved and/or during ranking.

Non-textual documents

Some document collections may be non-textual

- Documents can be images, movies, sound files etc.
- This situation gives rise to *Multimedia Information Retrieval*.
- An emerging, increasingly important field.

Document pointers and indices

Web documents are usually considered as text documents, possibly with many fields (text, title, anchors [careful! anchors of document *d* are contained in other documents]).

In this brief IR primer, we shall therefore stick to the traditional view in which a document is a piece of text.

- So $\mathcal{D}$ is a collection of text documents, each identified with a number (called its *document pointer* or docId).
- To answer the queries, typically one relies on the help of addictional data structures (built once at the beginning), called *indices*.
- What exactly is an index depends on the type of query language and ranking used (more about this later).

Document crunching

- Documents must first be *segmented*, i.e., divided into *words*.
  - Language-dependent; non-trivial in some languages where word boundaries are loose.
  - What should we do about non-word constituents (e.g., punctuation, spaces, HTML tags. . .)? Throw them away? Keep for later use? (Snippetting?)
- Words are typically *normalized*
  - Basic normalization: case folding, truecasing, spelling adjustments/corrections etc.
  - Possibly, stemming or lemmatization (took $\rightarrow$ tak-, taking $\rightarrow$ tak-)
  - Possibly, enriched with NLP information (Washington $\rightarrow$ Washington$\langle$LOCATION$\rangle$, Washington $\rightarrow$ Washington$\langle$PERSON$\rangle$)
  - Stopwords may be eliminated
  - What to do about *hapax legomena*?

Terms and inverted indices

After processing all documents, we end up with a set of *terms* that form a *vocabulary* $\mathcal{V}$.

- A basic concept in IR is the idea of *inverted list*.
- The inverted list of term $t \in \mathcal{V}$ is the list $IL(t)$ of all document pointers of documents where t appears, in increasing order.
- E.g., $IL(\text{dog}) = \langle 3, 12, 13, 45, 56, 77 \rangle$ (the word "dog" appears in the documents 3, 12, etc.)
- The set of all inverted lists is called an *inverted index*.
- More about how inverted indices are built in practice, later.

More terminology

We define (for $v \in \mathcal{V}$ and $d \in \mathcal{D}$)

- N: the number of documents
- ℓ_d: the length of document d
- L: the collection length $(= \sum_d \ell_d)$
- $\hat{\ell}$: the average document length $(= L/N)$
- df_v: the *document frequency* of v (number of documents where v appears)
- $\mathrm{tf}_{d,v}$: the *term frequency* of v in d (number of times v appears in d)
- cf_v: the *collection frequency* of v $(=\sum_d \mathrm{tf}_{dv})$

Interlude: IR evaluation

Interlude: IR evaluation

Ground truth

Suppose that, for each $q \in \mathcal{Q}$ an *omniscent being* can determine which set $R_q \subseteq \mathcal{D}$ are *relevant for q*. We are supposing that every document is either relevant or irrelevant.

Evaluating unranked engines

An unranked IR engine can be seen as a function

$$f : \mathcal{Q} \to 2^{\mathcal{D}}$$

where $f(q)$ is the set of documents returned by the system for the query q.

- *Precision:* $P = |R_q \cap f(q)|/|f(q)|$ (percentage of retrieved documents that are relevant)
- *Recall:* $R = |R_q \cap f(q)|/|R_q|$ (percentage of relevant documents that are retrieved)
- *Accuracy:* $A = 1 - |(R_q \Delta f(q))|/N$ (percentage of relevant documents retrieved or irrelevant documents nonretrieved).
- *F-measure:* weighted harmonic mean between P and R

Evaluating ranked engines

A ranked IR engine can be seen as a function

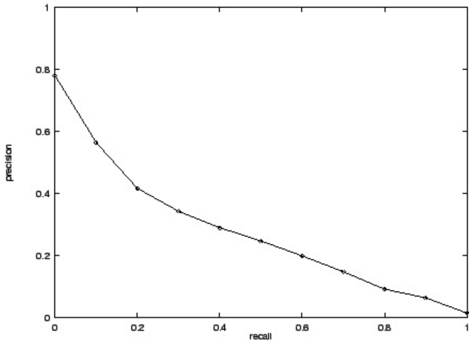
$$f : \mathcal{Q} \to \pi(\mathcal{D})$$

that, for a given query q, produces an ordering $f(q)$ of the documents, say $d_{f(q,1)}, d_{f(q,2)}, \dots, d_{f(q,r_q)}$, where $d_{f(q,1)}$ is the most relevant document and r_q is the number of documents retrieved.

The same measures used for the unranked case can be adapted to the ranked case by considering as "retrieved" the set of top k documents retrieved by the system. So we have $P@k$ (precision at k) and $R@k$ (recall at k).

Precision/recall curve

One can consider, for every value $r \in [0,1]$ of recall, what is the number of documents needed to obtain that recall (i.e., the smallest k such that $r \geq R@k$) and look at the precision at that level. This produces the so call *precision/recall* curve (usually evaluated at the 11 points of recall $0, 0.1, \ldots, 0.9, 1.0$).



Mean average precision

Another measure is the following:

- let $d_1, \dots, d_k$ be the *relevant retrieved* documents in the order in which they are output
- let $P(i)$ be the precision when d_i is output
- the average of the $P(i)$'s is called the MAP (Mean Average Precision).

…and many many many more…

Boolean model of IR

Boolean model of IR

Boolean query resolution

Under the *boolean query* assumptions, the query language allows one to use standard boolean operators (AND, OR, NOT).

- Query processing corresponds to retrieving the inverted list of the terms and merging them (by intersection, for AND, by union, for OR). E.g., "dog AND cat" is solved by intersecting IL(dog) wit IL(cat).
- Merging two posting lists of length x and y time $O(x + y) = O(L)$.
- Merging three posting lists of length x_1, x_2, x_3, requires $O(x_1 + x_2 + r_{12} + x_3)$ where r_{12} is the size of intersecting the first two lists.
- Good heuristic: solve by increasing document frequency.
- When AND/OR have many terms, use a heap and solve them altogether.
- Having a skip structure in the index helps with AND queries.

Enriching query language

Enriching the query language with other operators (proximity, phrase etc.) requires adding some information (in particular, positions) to the inverted index:

$$IL(\text{dog}) = \langle 3 : (70, 73, 150), 12 : (13, 45), 13 : (53, 54, 58), \dots \rangle$$

In alternative, one may adopt *biword indices* (indexing pairs of consecutive words as terms).

Vector space model of IR

Vector space model of IR

Vector space representation of objects

A general problem:

- You have a set O of objects (e.g., images)
- You have a set F of features (e.g., prevalent color, brightness, etc.)
- For each object $x \in O$ and each feature $f \in F$, you may "measure" f in x; this measurement provides a number (0/1 or a real number of some sort)
- So you can identify every $x \in O$ with a vector $\mathbf{x} \in \mathbf{R}^F$: the entry x_f is the measure of f on x.

This is called the vector space representation of O with respect to the set of features F.

Cosine distance

Now, given two objects (i.e., vectors) $\mathbf{x}, \mathbf{y} \in \mathbf{R}^F$, it is known that their dot-product satisfies

$$\mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\| \|\mathbf{y}\| \cos(\vartheta)$$

where $\| - \|$ represents their L2 norm ($\|\mathbf{x}\| = \sqrt{\mathbf{x} \cdot \mathbf{x}}$) and ϑ is the angle formed by the two vectors.

As a consequence

$$\cos(\vartheta) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}.$$

This is called the *cosine distance* (or cosine similarity) between $\mathbf{x}$ and $\mathbf{y}$: it is 1 iff the two vectors are parallel, and 0 if they are orthogonal.

Example: binary representation of a document

Suppose you want to apply the vector space model to documents. Then you can represent a document $d \in \mathcal{D}$ with a vector $\mathbf{x}(d) \in \mathbf{R}^{\mathcal{V}}$ having one entry per term: the entry relative to term $t \in \mathcal{V}$ measures the presence of t in d.

A brutal way to do that is to use a binary vector, letting

$$x(d)_t = \begin{cases} 1 & \text{if } t \text{ appears in } d \\ 0 & \text{if } t \text{ does not appear in } d. \end{cases}$$

In this case the cosine distance between d and d' is

$$\frac{\mathbf{x}(d) \cdot \mathbf{x}(d')}{\|\mathbf{x}(d)\| \|\mathbf{x}(d')\|}.$$

This may be called the *binary vector model* for documents.

Example: binary representation of a document (cont'd)

Now let $q \in \mathcal{Q}$ be a query (a *bag-of-words*) and $d \in \mathcal{D}$ be a document. Define the score of document d for query q as their cosine similarity, i.e.,

$$\frac{\mathbf{x}(d) \cdot \mathbf{x}(q)}{\|\mathbf{x}(d)\| \|\mathbf{x}(q)\|}.$$

Documents will be ranked by decreasing score. Equivalently, since $\|\mathbf{x}(q)\|$ is constant, they can be ranked by

$$\frac{\mathbf{x}(d) \cdot \mathbf{x}(q)}{\|\mathbf{x}(d)\|}.$$

The numerator is just $|d \cap q|$ (the number of words of the query that appear in the document), wheras the denominator is just the (square root of) the length of document d.

Extension: using term frequencies

The binary representation does not take into account the number of times a term appears in a document, but this can be simply solved by letting

$$x(d)_t = \mathrm{tf}_{t,d}.$$

This is the number of times that t does not appears in d.

As before, you can rank documents with respect to a query $q \in \mathcal{Q}$ by

$$\frac{\mathbf{x}(d) \cdot \mathbf{x}(q)}{\|\mathbf{x}(d)\|},$$

or explicitly

$$\frac{\sum_{t \in q} \mathrm{tf}_{t,d}}{\sqrt{\sum_{t \in d} \mathrm{tf}_{t,d}}} = \sum_{t \in q} \widehat{\mathrm{tf}}_{t,d}$$

where $\widehat{\mathrm{tf}}$ represents the term frequency after Euclidean normalization.

Drawbacks

The formula we obtained

$$\sum_{t \in q} \widehat{\mathrm{tf}}_{t,d}$$

has a drawback if q contains some very common word (like "the" or "of"): the score will get very high on documents that satisfy the query just because they contain many occurrences of useless words!

More generally: some terms should be considered more important than others; rare terms are such. With this purpose, we introduce an attenuation factor that depends on the document frequency of a term (i.e., the number of documents where it appears):

$$\sum_{t \in q} \widehat{\mathrm{tf}}_{t,d} \mathrm{idf}_t$$

where

$$\mathrm{idf}_t = \log \frac{N}{\mathrm{df}_t} \quad \textit{inverse document frequency.}$$

Probabilistic model of IR

Probabilistic model of IR

Idea behind the probabilistic model

The idea behind this model is that *being relevant* can be thought of as a random variable Rel, and that the document and the query themselves are random variables **D** and **Q**: to score a document we must compute $P[\mathrm{Rel} = 1 \mid \mathbf{D} = \mathbf{d}, \mathbf{Q} = \mathbf{q}]$ (the probability that the document is relevant given the values of the document and the query).

In this formulation, Rel is a binary random variable (it can take only values 0=irrelevant and 1=relevant), whereas **Q** and **D** depend on how we want to represent queries and documents. They can be vectors of term frequency, or some other set of features that we want to use to represent the query and the document.

If we use document frequency, $d_t = \mathrm{tf}_{t,d}$. In general d_t is some feature (a number or something more complex) of t in d.

Idea behind the probabilistic model

From now on, we use rel ($\overline{\mathrm{rel}}$) as a short form for the event $\mathrm{Rel} = 1$ ($\mathrm{Rel} = 0$, respectively). Moreover we write **q** and **d** as a short form for $\mathbf{Q} = \mathbf{q}$ and $\mathbf{D} = \mathbf{d}$.

So we want to rank documents according to (in decreasing order of)

$$P[\mathrm{rel} \mid \mathbf{q}, \mathbf{d}].$$

The actual estimation of this ranking can be accomplished in many different ways. We will show how to derive the basic scoring formula that it yields

We use $\approx$ for approximated equality (usually, under some assumptions) and $\bowtie$ for "scores documents in the same order as" (rank-equivalence).

Probabilistic model

$$\begin{aligned}
P[\mathrm{rel} \mid \mathbf{q}, \mathbf{d}] &\bowtie \frac{P[\mathrm{rel} \mid \mathbf{q}, \mathbf{d}]}{P[\overline{\mathrm{rel}} \mid \mathbf{q}, \mathbf{d}]} && \text{odds-ratio} \\
&= \frac{P[\mathbf{d}, \mathbf{q} \mid \mathrm{rel}]}{P[\mathbf{d}, \mathbf{q} \mid \overline{\mathrm{rel}}]} \cdot \frac{P[\mathrm{rel}]}{P[\overline{\mathrm{rel}}]} && \text{Bayes formula} \\
&= \frac{P[\mathbf{d} \mid \mathbf{q}, \mathrm{rel}] P[\mathbf{q} \mid \mathrm{rel}]}{P[\mathbf{d} \mid \mathbf{q}, \overline{\mathrm{rel}}] P[\mathbf{q} \mid \overline{\mathrm{rel}}]} \cdot \frac{P[\mathrm{rel}]}{P[\overline{\mathrm{rel}}]} && \text{chain rule} \\
&\bowtie \frac{P[\mathbf{d} \mid \mathbf{q}, \mathrm{rel}]}{P[\mathbf{d} \mid \mathbf{q}, \overline{\mathrm{rel}}]} && \text{drop parts indep. of } d
\end{aligned}$$

Probabilistic model — Term conditional independence

To go on from here, we need some assumptions. The first is called *Term conditional independence*:

- in our model, the terms that appear in a document (and their frequency) are independent of one another, although they may depend on the query and relevance;
- in other words, if we fix a query **q** and a relevance Rel, the probability that a document with relevance Rel with respect to **q** contains any given term with any given frequency is independent of the other terms appearing in the document:

$$\begin{aligned} P[\mathrm{rel} \mid \mathbf{q}, \mathbf{d}] &\bowtie \frac{P[\mathbf{d} \mid \mathbf{q}, \mathrm{rel}]}{P[\mathbf{d} \mid \mathbf{q}, \overline{\mathrm{rel}}]} \\ &\approx \prod_{t \in \mathcal{V}} \frac{P[d_t = \mathrm{tf}_t \mid \mathbf{q}, \mathrm{rel}]}{P[d_t = \mathrm{tf}_t \mid \mathbf{q}, \overline{\mathrm{rel}}]} \qquad \text{term cond. independence} \end{aligned}$$

Probabilistic model — Query term assumption

The next assumption is called *Query term assumption*:

- *Query-term assumption:* that relevance depends only on the terms in the query and not on any other term of the document.

$$
\begin{aligned}
P[\mathrm{rel} \mid \mathbf{q}, \mathbf{d}] &\bowtie \prod_{t \in \mathcal{V}} \frac{P[d_t = \mathrm{tf}_t \mid \mathbf{q}, \mathrm{rel}]}{P[d_t = \mathrm{tf}_t \mid \mathbf{q}, \overline{\mathrm{rel}}]} \\
&\approx \prod_{t \in \mathbf{q}} \frac{P[d_t = \mathrm{tf}_t \mid \mathrm{rel}]}{P[d_t = \mathrm{tf}_t \mid \overline{\mathrm{rel}}]} \qquad \text{query-term assumption} \\
&\bowtie \sum_{t \in \mathbf{q}} \log \frac{P[d_t = \mathrm{tf}_t \mid \mathrm{rel}]}{P[d_t = \mathrm{tf}_t \mid \overline{\mathrm{rel}}]} \qquad \text{logarithm}
\end{aligned}
$$

Probabilistic model — The zero trick

So

$$P[\mathrm{rel} \mid \mathbf{q}, \mathbf{d}] \bowtie \sum_{t \in \mathbf{q}} \log \frac{P[d_t = \mathrm{tf}_t \mid \mathrm{rel}]}{P[d_t = \mathrm{tf}_t \mid \overline{\mathrm{rel}}]}.$$

Defining

$$U_t(x) = \log \frac{P[d_t = x \mid \mathrm{rel}]}{P[d_t = x \mid \overline{\mathrm{rel}}]}$$

we have

$$P[\mathrm{rel} \mid \mathbf{q}, \mathbf{d}] \bowtie \sum_{t \in \mathbf{q}} U_t(d_t).$$

We break this summation distinguishing the terms that appear in the document ($d_t > 0$) from the others and apply a trick:

$$\begin{aligned} P[\mathrm{rel} \mid \mathbf{q}, \mathbf{d}] &\bowtie \sum_{t \in \mathbf{q}, d_t > 0} U_t(d_t) + \sum_{t \in \mathbf{q}, d_t = 0} U_t(0) + \sum_{t \in \mathbf{q}, d_t > 0} U_t(0) - \sum_{t \in \mathbf{q}, d_t > 0} U_t(0) \\ &= \sum_{t \in \mathbf{q}, d_t > 0} (U_t(d_t) - U_t(0)) + \sum_{t \in \mathbf{q}} U_t(0) \bowtie \sum_{t \in \mathbf{q}, d_t > 0} (U_t(d_t) - U_t(0)). \end{aligned}$$

Probabilistic model — Finally

So finally

$$P[\mathrm{rel} \mid \mathbf{q}, \mathbf{d}] \bowtie \sum_{t \in \mathbf{q} \cap \mathbf{d}} w_{t,d}.$$

where

$$\begin{aligned} w_{t,d} &= W_t(d_t) := U_t(d_t) - U_t(0) \\ W_t(x) &= \log \frac{P[\mathrm{tf}_{t,d} = x \mid \mathrm{rel}] P[\mathrm{tf}_{t,d} = 0 \mid \overline{\mathrm{rel}}]}{P[\mathrm{tf}_{t,d} = x \mid \overline{\mathrm{rel}}] P[\mathrm{tf}_{t,d} = 0 \mid \mathrm{rel}]}. \end{aligned}$$

As a result, the score of a document simply consists in summing the "weights" of some of the terms appearing in the document (precisely, of the terms that also happen to appear in the query as well).

Note that those weights depend on the term and on the document, but not on the query! They may be stored in the inverted list: for every term t and every document d where t appears, we store $w_{t,d}$.

We then can solve the query in the boolean way, taking the union of the inverted lists of the terms in the query, and summing the weights of the document in the intersection.

So...

All in all, we have to compute, for every term t and every document d, a weight $w_{t,d}$ (the weight for the term t in document d), whose value is given by $W_t(\mathrm{tf}_{t,d})$, with

$$W_t(x) = \log \frac{P[\mathrm{tf}_{t,d} = x \mid \mathrm{rel}]P[\mathrm{tf}_{t,d} = 0 \mid \overline{\mathrm{rel}}]}{P[\mathrm{tf}_{t,d} = x \mid \overline{\mathrm{rel}}]P[\mathrm{tf}_{t,d} = 0 \mid \mathrm{rel}]}.$$

How this quantity is estimated depends on the actual probabilistic model adopted, and gives raise to different concrete weighting formulas.

The most well-known is the so-called *BM25* (a.k.a. Okapi):

$$W_t^{\mathrm{BM25}}(x) = \frac{x}{x + k1 \cdot (1 - b + \ell_d/\hat{\ell})} \cdot \log \frac{N - x + 0.5}{x + 0.5}$$

where $k_1 \in [1.2, 2.0]$ and $b \in [0.7, 0.8]$ are two parameters.

How BM25 is derived

For a fixed document d, every term t appearing in the document can be classified as *elite* or *non-elite*: elite terms are those that correspond to concepts describing the topic of the document, whereas non-elite are the other ones. Let $e_{t,d}$ denote the fact that t is elite in d.

Let $E_{t,e}$ be a random variable describing the frequency of t in a document, provided that it is elite (if $e = 1$) or non-elite (if $e = 0$). We assume that $E_{t,e}$ is distributed as a Poisson with mean $\lambda_{t,e}$: of course, we expect that $\lambda_{t,1} > \lambda_{t,0}$.

So, for example,

$$\begin{aligned} P[\mathrm{tf}_{t,d} = x \mid \mathrm{rel}] &= \\ P[\mathrm{tf}_{t,d} = x \mid e_{t,d}]P[e_{t,d} \mid \mathrm{rel}] &+ P[\mathrm{tf}_{t,d} = x \mid \text{not } e_{t,d}]P[\text{not } e_{t,d} \mid \mathrm{rel}] \\ &= E_{t,1}(x)p_{t,1} + E_{t,0}(x)p_{t,0} \end{aligned}$$

where $p_{t,e}$ is the probability that t is elite (or non-elite) in a relevant document, and $E_{t,x}$ is a Poisson with parameter $\lambda_{t,x}$.

How BM25 is derived (cont'd)

Now, in theory we may substitute this and the other terms in the general formula of $W_t(x)$, obtaining a function $W_t^{\text{eliteness}}$ that would depend on the λ's and on the p's. This function, however, depends on far too many parameters and would anyway be extremly involved.

Nonetheless, some things will certainly hold:

- $W_t^{\text{eliteness}}(0) = 0$
- $W_t^{\text{eliteness}}(x)$ is monotonically increasing, with a horizontal asymptote as $x \to \infty$ (saturation)
- the saturation value is

$$s_t = \log \frac{p_{t,1}(1 - p_{t,0})}{(1 - p_{t,1})p_{t,0}}$$

Saturation

Let us concentrate on the value at saturation:

$$s_t = \log \frac{p_{t,1}(1 - p_{t,0})}{(1 - p_{t,1})p_{t,0}}$$

Recall that $p_{t,1}$ is the probability that t is elite in a relevant document, that we may compute as the fraction of relevant document that contain t. (Same for $p_{t,0}$)

Supposed that we sampled S documents and s_t of them contain t; suppose further that R of the documents are relevant and r_t of them contain t. We might estimate s_t:

$$s_t \approx \log \frac{\frac{r_t}{R}\left(1 - \frac{s_t - r_t}{S - R}\right)}{\left(1 - \frac{r_t}{R}\right)\frac{s_t - r_t}{S - R}}$$

Saturation (cont'd)

The formula simplifies to

$$s_t \approx \log \frac{r_t(S - R - s_t + r_t)}{(R - r_t)(s_t - r_t)}$$

and to avoid the treatment of special cases, we smooth it to

$$s_t \approx \log \frac{(r_t + 0.5)(S - R - s_t + r_t + 0.5)}{(R - r_t + 0.5)(s_t - r_t + 0.5)}.$$

Since we don't have any judgement, $R = r_t = 0$, $S = N$ and $s_t = \mathrm{df}_t$:

$$s_t \approx \log \frac{N - \mathrm{df}_t + 0.5}{\mathrm{df}_t + 0.5}$$

How BM25 is derived (cont'd)

Getting back to our original problem, we need to approximate $W_t^{\mathrm{eliteness}}$ with a function W_t^{BM25} such that:

- $W_t^{\mathrm{BM25}}(0) = 0$
- $W_t^{\mathrm{BM25}}(x)$ is monotonically increasing, with a horizontal asymptote as $x \to \infty$ (saturation) with value

$$\log \frac{N - \mathrm{df}_t + 0.5}{\mathrm{df}_t + 0.5}.$$

Looking at the shape of $W_t^{\mathrm{eliteness}}$, we come to

$$W_t^{\mathrm{BM25}}(x) = \frac{x}{x + k_1} \cdot \log \frac{N - \mathrm{df}_t + 0.5}{\mathrm{df}_t + 0.5},$$

where k_1 is a suitable parameter.

How BM25 is derived (cont'd)

The final formula obtained

$$W_t^{\mathrm{BM25}}(x) = \frac{x}{x + k_1} \cdot \log \frac{N - \mathrm{df}_t + 0.5}{\mathrm{df}_t + 0.5},$$

needs to be adjusted for document lengths, obtaining the final formula

$$W_t^{\mathrm{BM25}}(x) = \frac{x}{x + k_1(1 - b + \ell_d/\hat{\ell})} \cdot \log \frac{N - \mathrm{df}_t + 0.5}{\mathrm{df}_t + 0.5},$$