


Recuperación de la Información

Ana X. Ezquerro

ana.ezquerro@udc.es,  GitHub

Grado en Ciencia e Ingeniería de Datos
Universidad de A Coruña (UDC)

Curso 2021-2022

Table of Contents

1. Search Engine Architecture	3
1.1. <i>The anatomy of a large-scale hypertextual Web search engine</i>	3
1.2. Brief history	3
1.3. Spam	4
1.4. Advertising as the economic model	4
2. Basic Retrieval Models	6
2.1. Boolean Retrieval Model	6
2.2. Term frequency and weighting	6
3. Probabilistic Retrieval Models	7
3.1. Probability Ranking Principle	7
3.2. Binary Independence Model (BIM)	7
3.3. Okapi BM25. A nonbinary model	9
4. Language Models	10
4.1. Query likelihood model	10
4.2. Smoothing	10

Chapter 1: Search Engine Architecture

1.1. The anatomy of a large-scale hypertextual Web search engine

- Google: Prototype of a large-scale search engine, which makes heavy use of the structure present in hypertext.
- Google is designed to crawl and index the Web efficiently and produce much more satisfying search results.
- Yahoo!: High quality human maintained indices.
- Automated search engines that rely on keyword matching usually return too many low quality matches.
- Some advertisers attempt to gain attention by taking measures meant to mislead automated search engines.
- Fast crawling technology is needed to gather the Web documents and keep them up to date.
- In designing Google, the rate of growth of the Web and technological changes have been considered.
- Efficient use of storage space to store the index.
- Data structures are optimized for fast and efficient access.
- Tools with very high precision are needed.
- Makes use of the link structure of the Web to calculate a quality ranking for each Web page (**PageRank**).
- Utilizes links to improve search results.
- PageRank counts citations or backlinks to a given page and normalizes by the number of links on a page.

PageRank

Assume page A has T_1, \dots, T_n citations which point to it. Let $d \in [0, 1]$ be a damping factor and $C(A)$ the number of links going out of page A . The PageRank of page A is given as:

$$\text{PageRank}(A) = (1 - d) + d \left(\frac{\text{PageRank}(T_1)}{C(T_1)} + \dots + \frac{\text{PageRank}(T_n)}{C(T_n)} \right)$$

- Pagerank corresponds to the principal eigenvector of the normalized link matrix of the Web.

1.2. Brief history

Early attempts at making web information “discoverable” fell into two broad categories:

- Full-text index search engines (such as Altavista, Excite, Infoseek, Inktomi, Lycos) which presented the user with a keyword search interface supported by inverted indexes and ranking mechanisms.
- Taxonomies populated with web pages in categories (such as Yahoo!), which allowed the user to browse through a hierarchical tree of category labels.

The first generation of web search engines transported classical search techniques, focusing on the challenge of scale. However, the quality and relevance of web search results left much to be desired owing to the idiosyncrasies of content creation on the Web.

- Need of new ranking and spam-fighting techniques in order to ensure the quality of search results.
- Classical techniques measure the relevance of a document to a query.
- Need to gauge the *authoritativeness* of a document based on cues such as which website hosts it.
- Web search engines were an important means for connecting advertisers to prospective buyers.
- *Paid search* ranking (e.g. Goto)

Since 1998, Google pioneered *link-based ranking*, which blew away all early engines thanks to great user experience in search.

Google added *paid search* “ads” to the side, independent of search results. Yahoo followed suit, acquiring Overture (for paid placement) and Inktomi (for search).

Since 2005, Google gains search share, dominating in Europe and North America.

1.3. Spam

Manipulation of web page content for the purpose of appearing high up in search results for selected keywords.

- *Cloacking*: Spammer's web server returns different pages depending on whether the http request comes from a web search engine's crawler or from a human user's browser. The former causes the web page to be indexed by the search engine under misleading keywords.
- *Doorway page*: Content text and metadata carefully chosen to rank highly.

To combat spammers who manipulate the text of their web pages is the exploitation of the link structure of the Web (link analysis).

1.4. Advertising as the economic model

1.4.1. First generation

- Companies used graphical banner advertisements on web pages at popular websites.
- These advertisements are priced on a *cost per mil* (CPM) basis: the cost to the company of having its banner advertisement displayed 1000 times.
- *Cost per click* (CPC) model: Advertisements are priced by the number of times it is clicked on by the user, so the goal of the advertisements is to induce a *transaction*. Clicks could be metered and monitored by the website and billed to the advertiser.
- **Goto** (*sponsored search* or *search advertising*): For every query term q it accepted *bids* from companies who wanted their web page shown on the query q . When the user clicked on one of the returned results, the corresponding advertiser would make a payment to Goto.
 - Only on results list.
 - Initial implementation: the payment equaled the advertiser's bid for q .

1.4.2. Second generation

- Provide pure search results (*algorithmic search* results) as the primary response to a user's search with *sponsored search* results displayed separately.

- *Click spam*: Clicks on sponsored search results that are not from bonafide search users.

1.4.3. How are ads ranked? Second price auction

Basis:

- Advertisers bid for keywords (sale by auction) and they are only charged when somebody clicks on their ad.
- How does this auction determine ad's rank and the price paid?
- First cut: Rank = bid price (bad idea since it is not desirable to show nonrelevant ads).

Idea: Rank based on bid price and relevance.

- Key measure of ad relevance: **click through rate** (CTR), i.e. clicks per impressions.
- Hope: Overall acceptance of the system and overall revenue is maximized if users get useful information.
- Other ranking factors: location, time of day, quality and loading speed of landing page...

Parameters:

- bid: Maximum bid for a click by advertiser.
- CTR: What percentage of times do users click on ad?
- ad rank = bid \times CTR.
- paid: Minimum amount necessary to maintain their position in the auction (plus 1 cent).

$$\text{paid}_1 \times \text{CTR}_1 = \text{bid}_2 \times \text{CTR}_2$$

$$\text{paid}_1 = \text{bid}_2 \times \text{CTR}_2 / \text{CTR}_1$$

Advantages:

- The search engine company gets revenue every time somebody clicks on ad.
- Users only clicks on an ad they are interested in.

- Search engines punish misleading and nonrelevant ads. As a result, users are often satisfied.
- The advertiser finds new customers in a cost-effective way.

Problems:

- Keyword arbitrage: Spammers buy a keyword on Google and redirect traffic to a third party that is paying much more than they are paying Google.
- Violation of trademarks. It's potentially misleading to users to trigger an ad off of a trademark if they can't buy the product on the site.

1.4.4. Third generation: Real Time Bidding

- An advertiser submits a bid for each individual impression in a very short time frame, often less than 100ms.
- Per impression transactions scales the process across a large number of available ad inventories.
- Real time audience data encourages behaviour (re)targeting shifting towards the use of **user data** rather than contextual data.

Chapter 2: Basic Retrieval Models

2.1. Boolean Retrieval Model

Given the query with two terms $q = x_1 \text{AND} x_2$:

1. Locate x_1 in the dictionary and retrieve its postings p_1 .
2. Locate x_2 in the dictionary and retrieve its postings p_2 .
3. Intersect the two posting lists.

The *intersection* operation of two posting lists (also known as *merging*) can be computed in $O(|p_1| + |p_2|)$ using a numeric sort by docID. In case of having more than two posting lists, it is more efficient to intersect each retrieved posting list with the current intermediate result in memory. A trick for speed-up is to start from posting list with the lowest number of documents to the highest.

2.2. Term frequency and weighting

Motivation:

- Both query and document can be seen as a set of terms.
- The score of a document is highly correlated with the presence/absence of a query term in it.

A form of weighting a document term (in order to compute a score for the document) is to use the number of occurrences of a term t in the document d , i.e. use the **term frequency**:

$$\text{tf}_{td} = \text{number of occurrences of term } t \text{ in document } d$$

Another weighting scheme for document terms is to use the **document frequencies** to assign higher weights to discriminative terms in the query. Let df_t be the number of document in the collection that contain term t . Each term t can be weighted with the **inverse document frequency**:

$$\text{idf}_t = \log \frac{N}{\text{df}_t}$$

where N is the number of documents in the collection.

If both definitions are combined (term frequency with inverse document frequency) we have the **tf-idf weighting** scheme, where each term t in a document d is weighted by:

$$\text{tf-idf}_{td} = \text{tf}_{td} \cdot \text{idf}_t$$

which is

- highest when t occurs many times within a small number of documents.
- lower when the term occurs fewer times in a document or occurs in many documents.
- lowest when the term occurs in virtually all documents.

This way we can define the score of a document d given a query q with:

$$\text{score}(q, d) = \sum_{t \in q} \text{tf-idf}_{td}$$

2.2.1. Variants of tf-idf functions

- **Sublinear tf scaling**: The number of occurrences of a term t in a document d is not linearly correlated with the level of significance.

$$\text{wtf}_{td} = \begin{cases} 1 + \log \text{tf}_{td} & \text{tf}_{td} > 0 \\ 0 & \text{otherwise} \end{cases}$$

- **Maximum tf normalization**. Normalize the term frequency of a term t by the maximum tf in that document d :

$$\text{ntf}_{td} = \alpha + (1 - \alpha) \frac{\text{tf}_{td}}{\text{tf}_{\max}(d)}$$

where $\alpha \in [0, 1]$ (normally 0.4) is a smoothing term to damp the contribution of the second term.

Chapter 3: Probabilistic Retrieval Models

3.1. Probability Ranking Principle

Let's assume a binary notion of relevance. For a query q and a document d in the collection, let $R_{d,q}$ or R be an indicator random variable that says whether d is relevant with respect to a given query q .

The **Probability Ranking Principle** (PRP) says that, if the response to a request is a ranking of the documents in the collection in order of decreasing probability of relevance, where probabilities are estimated as accurately as possible, then the overall effectiveness of the system will be the best that is obtainable.

- Simplest case: There are no retrieval costs, i.e. you are evaluated on your *accuracy*, what is called 1/0 loss.
 - PRP simply ranks all documents in decreasing order of $\mathbb{P}(R = 1|d, q)$.
 - If a set is needed, use the *Bayes optimal decision rule* that minimizes the risk of loss.

$$\text{return } d \iff \mathbb{P}(R = 1|d, q) > \mathbb{P}(R = 0|d, q)$$

- PRP with retrieval costs. Let C_1 be the cost of retrieval of a relevant document and C_0 the cost of retrieval a nonrelevant document. PRP returns a document d given all documents d' not yet retrieved if and only if:

$$C_1 \cdot \mathbb{P}(R = 1|d) + C_0 \cdot \mathbb{P}(R = 0|d) \leq C_1 \cdot \mathbb{P}(R = 1|d') + C_0 \cdot \mathbb{P}(R = 0|d')$$

3.2. Binary Independence Model (BIM)

Goal: Estimating probability function $\mathbb{P}(R|d, q)$.

Assumptions:

1. Documents and queries are represented as binary term incidence vectors.
2. Terms are modeled as occurring in documents independently (**Naive Bayes assumption**).
3. Relevance of each document is independent of relevance of other document.

BIM models the probability $\mathbb{P}(R|d, q)$ via the probability in terms of term incidence vectors.

$$\mathbb{P}(R = 1|\vec{x}, \vec{q}) = \frac{\mathbb{P}(\vec{x}|R = 1, \vec{q}) \cdot \mathbb{P}(R = 1|\vec{q})}{\mathbb{P}(\vec{x}|\vec{q})}$$

$$\mathbb{P}(R = 0|\vec{x}, \vec{q}) = \frac{\mathbb{P}(\vec{x}|R = 0, \vec{q}) \cdot \mathbb{P}(R = 0|\vec{q})}{\mathbb{P}(\vec{x}|\vec{q})}$$

Since BIM is only interested in the ranking of documents, it uses the odds of relevance so it can ignore the common denominator $\mathbb{P}(\vec{x}|\vec{q})$:

$$\text{odd}(R|\vec{x}, \vec{q}) = \frac{\mathbb{P}(R = 1|\vec{x}, \vec{q})}{\mathbb{P}(R = 0|\vec{x}, \vec{q})} = \frac{\mathbb{P}(\vec{x}|R = 1, \vec{q})}{\mathbb{P}(\vec{x}|R = 0, \vec{q})} \cdot \underbrace{\frac{\mathbb{P}(R = 1|\vec{q})}{\mathbb{P}(R = 0|\vec{q})}}_{\text{ignored constant}}$$

With the *Naive Bayes conditional independence assumption*, $\mathbb{P}(\vec{x}|R, \vec{q})$ can be estimated via the independent presence of terms x_t , for $t = 1, \dots, M$.

$$\text{odd}(R|\vec{x}, \vec{q}) = \text{odd}(R|\vec{q}) \cdot \prod_{t=1}^M \frac{\mathbb{P}(x_t|R = 1, \vec{q})}{\mathbb{P}(x_t|R = 0, \vec{q})}$$

Because each $x_t \in \{0, 1\}$, terms can be separated to give:

$$\text{odd}(R|\vec{x}, \vec{q}) = \text{odd}(R|\vec{q}) \cdot \prod_{t=1}^M \frac{\mathbb{P}(x_t = 1|R = 1, \vec{q})}{\mathbb{P}(x_t = 1|R = 0, \vec{q})} \cdot \prod_{t=1}^M \frac{\mathbb{P}(x_t = 0|R = 1, \vec{q})}{\mathbb{P}(x_t = 0|R = 0, \vec{q})}$$

Let

- $p_t = \mathbb{P}(x_t = 1|R = 1, \vec{q})$ be the probability of a term appearing in a relevant document.
- $u_t = \mathbb{P}(x_t = 1|R = 0, \vec{q})$ be the probability of a term appearing in a nonrelevant document.
- $1 - p_t = \mathbb{P}(x_t = 0|R = 1, \vec{q})$ be the probability of a term not appearing in a relevant document.

- $1 - u_t = \mathbb{P}(x_t = 0 | R = 0, \vec{q})$ be the probability of a term not appearing in a nonrelevant document.

Assumption: Terms not occurring in the query a are equally likely to occur in relevant and nonrelevant documents. If $q_t = 0$, then $p_t = u_t$.

Now BIM only considers terms that appear in the query:

$$\text{odd}(R|\vec{x}, \vec{q}) = \text{odd}(R|\vec{x}) \cdot \overbrace{\prod_{t:x_t=q_t=1} \frac{p_t}{u_t}}^{\text{query terms found in doc}} \cdot \underbrace{\prod_{t:x_t=0, q_t=1} \frac{1-p_t}{1-u_t}}_{\text{query terms not found in doc}}$$

This expression can be rewritten as:

$$\text{odd}(R|\vec{x}, \vec{q}) = \text{odd}(R|\vec{x}) \cdot \overbrace{\prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)}}^{\text{query terms found in doc}} \cdot \underbrace{\prod_{t:q_t=1} \frac{1-p_t}{1-u_t}}_{\text{query terms (constant)}}$$

Note: Only the left product is needed to compute a document ranking.

Instead of using the product of probabilities, the sum of logarithms can be used to compute the ranking. The resulting quantity is called the **Retrieval Status Value (RSV)**:

$$\text{RSV}_d = \sum_{t:x_t=q_t=1} \log \frac{p_t(1-u_t)}{u_t(1-p_t)}$$

Let c_t be the log odds ratios for the terms in the query:

$$c_t = \log \frac{p_t(1-u_t)}{u_t(1-p_t)} = \log \frac{p_t}{1-p_t} + \log \frac{1-u_t}{u_t}$$

- $c_t = 0$ if a term t has equal odds of appearing in relevant and nonrelevant documents.
- $c_t > 0$ if a term t is more likely to appear in relevant documents.

The c_t quantities function as term weights in the model.

$$\text{RSV}_d = \sum_{t:x_t=q_t=1} c_t$$

	documents	relevant	nonrelevant
term present	$x_t = 1$	p_t	u_t
term absent	$x_t = 0$	$1 - p_t$	$1 - u_t$

3.2.1. Probability estimates in theory

- df_t : Number of documents that contain term t .

	documents	relevant	nonrelevant	total
term present	$x_t = 1$	s	$\text{df}_t - s$	df_t
term absent	$x_t = 0$	$S - s$	$(N - \text{df}_t) - (S - s)$	$N - \text{df}_t$
total		S	$N - S$	N

We can estimate $p_t = s/S$ and $u_t = (\text{df}_t - s)/(N - S)$:

$$c_t = \log \frac{s/(S - s)}{(\text{df}_t - s)/[(N - \text{df}_t) - (S - s)]}$$

To avoid the possibility of zeroes, add $1/2$ (prior probability) to each of the estimation quantities of our probabilities and adjust the marginal counts accordingly (smoothing):

$$\hat{c}_t = \log \frac{(s + 0.5)/(S - s + 0.5)}{(\text{df}_t - s + 0.5)/(N - \text{df}_t - S + s + 0.5)}$$

3.2.2. Probability estimates in practice

Assumption: Relevant documents are a very small percentage of the collection, so it is plausible to approximate statistics for nonrelevant documents by statistics from the whole collection.

Thus, u_t (probability of term occurrence in nonrelevant documents) is df_t/N and:

$$\log \frac{1 - u_t}{u_t} = \log \frac{N - df_t}{df_t} \approx \log \frac{N}{df_t} \quad (3.1)$$

3.2.3. RSJ Model without relevance info

Assumptions:

- p_t is constant for all terms.
- All documents are nonrelevant (i.e. $S = 0$).

Now RSV can be recomputed as:

$$RSV_d = \sum_{t:x_t=q_t=1} c + \log \frac{N - df_t + 0.5}{df_t + 0.5}$$

3.3. Okapi BM25. A nonbinary model

From 3.1, the BM25 model improves the equation by factoring in the frequency of each term and document length:

$$RSV_d = \sum_{t \in q} \overbrace{\log \frac{N}{df_t}}^{\text{IDF}} \cdot \frac{(k_1 + 1)tf_{td}}{k_1[(1 - b) + b \cdot L_d/L_{ave}] + tf_{td}}$$

- tf_{td} : Frequency of term t in document d .
- L_d : Length of document d .
- L_{ave} : Average document length for the whole collection.
- $k_1 \in \mathbb{R}^+$: Tuning parameter that calibrates the document term frequency scaling.
 - $k_1 = 0$: Binary model (no term frequency).
 - $k_1 > 0$: Using term frequency.
 - Usually $k_1 \in [1.2, 2]$.
- $b \in [0, 1]$: Tuning parameter that determines the scaling by document length.

- $b = 1$: Fully scaling the term weight by document length.
- $b = 0$: No length normalization.
- Usually $b \in [0.75, 1.2]$.

For long queries, a similar weighting scheme might be used:

$$RSV_d = \sum_{t \in q} \overbrace{\log \frac{N}{df_t}}^{\text{IDF}} \cdot \frac{(k_1 + 1)tf_{td}}{k_1[(1 - b) + b \cdot L_d/L_{ave}] + tf_{td}} \cdot \frac{(k_2 + 1)tf_{tq}}{k_2 + tf_{tq}}$$

- tf_{tq} : Frequency of term t in the query q .
- k_2 : Tuning parameter that calibrates term frequency scaling of the query.

Chapter 4: Language Models

Motivation:

- Language Models provide a probabilistic mechanism to generate words.
- Given a Language Model build from a document d , M_d , compute the probability of generating the query q : $\mathbb{P}(q|M_d)$.

The most popular approach is to use the Unigram Model to estimate probabilities of word sequences $w_1 \dots w_n$:

$$\mathbb{P}(w_1 \dots w_n) = \prod_{i=1}^n \mathbb{P}(w_i)$$

4.1. Query likelihood model

Goal: Rank documents by $\mathbb{P}(d|q)$, where the probability of a document is interpreted as the likelihood that it is relevant to the query.

$$\mathbb{P}(d|q) = \frac{\overbrace{\mathbb{P}(q|d)}^{\text{assumed uniform}} \overbrace{\mathbb{P}(d)}^{\text{assumed uniform}}}{\underbrace{\mathbb{P}(q)}_{\text{same for } \forall d}}$$

Documents are ranked by the probability that a query would be observed as a random sample from the respective document model.

1. Infer a LM for each document d , M_d .
2. Estimate $\mathbb{P}(q|M_d)$, the probability of generating the query according to M_d .
3. Rank the documents according to these probabilities.

Using maximum likelihood assumption (MLE) and the unigram assumption, the estimation of this probability is:

$$\hat{\mathbb{P}}(q|M_d) = \prod_{t \in q} \hat{\mathbb{P}}_{\text{mle}}(t|M_d) = \prod_{t \in q} \frac{\text{tf}_{td}}{L_d}$$

4.2. Smoothing

Motivation: Any term that does not occur in the collection has zero probability with MLE. Our solution is to discount the probability of words seen in a document and re-allocate the extra counts in unseen words.

4.2.1. Additive smoothing

Add a constant δ to the counts of each word w in document d :

$$\mathbb{P}(w|d) = \frac{c(w, d) + 1}{L_d + |V|}$$

- $c(w, d)$: Count of word w in document d .
- V : Set of words in vocabulary.
- L_d : Document length.

4.2.2. Absolute discounting

Idea: Subtract a constant δ from the counts of each word.

$$\mathbb{P}(w|d) = \frac{\max(c(w, d) - \delta, 0) + \delta L_d^{(u)} \mathbb{P}(w|\text{ref})}{L_d}$$

- $L_d^{(u)}$: Number of unique words in document d .

4.2.3. Jelinek-Mercer smoothing

Idea: Linear interpolation of the MLE estimation with reference model using a coefficient λ to control the influence of each model.

$$\mathbb{P}(w|d) = (1 - \lambda) \frac{c(w, d)}{L_d} + \lambda \mathbb{P}(w|\text{ref})$$

