Master Degree in Computer Science

Information Retrieval

Probabilistic Information Retrieval

Prof. Alfio Ferrara

Department of Computer Science, Università degli Studi di Milano
Room 7012 via Celoria 18, 20133 Milano, Italia alfio.ferrara@unimi.it

sed noli modo

Motivations

La théorie des probabilités n'est, au fond, que le bon sens réduit au calcul;
elle fait apprécier avec exactitude ce que les esprits justes sentent par une sorte d'instinct, sans
qu'ils puissent souvent s'en rendre compte

[P.S. Laplace]

The complexity of text analysis is such that we cannot have a scientific law determining the correctness of an answer to a query

Thus we substitute the **True** with the **Probable**

Review of basic probability theory

Random variable

A variable A represents an **event**, a **subset** of the space of the **possible outcomes**

Another way of representing A is by a **random variable**, a **function from outcomes to real numbers**

Joint and conditional probabilities

We denote *P(A, B)* as the probability of two events A and B to occur together. We denote *P(A | B)* as the probability of A occurring given the occurrence of B

Chain rule

$$P(A, B) = P(A \cap B) = P(A \mid B)P(B) = P(B \mid A)P(A); \;\; P(\neg A, B) = P(\neg A \mid B)P(\neg A)$$

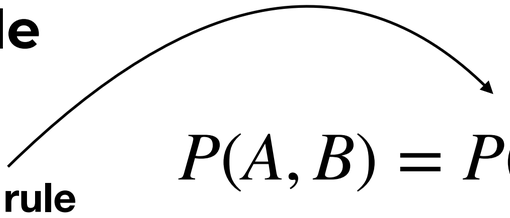
Review of basic probability theory

Partition rule

If an event B can be divided into an exhaustive set of disjoint sub-cases, then the probability of B is the sum of the probabilities of the sub-cases. Thus:

$$P(B) = P(A, B) + P(\neg A, B)$$

Bayes Rule



Chain rule

$$P(A, B) = P(A \mid B)P(B) = P(B, A) = P(B \mid A)P(A)$$

$$P(A \mid B)P(B) = P(B \mid A)P(A) \rightarrow \frac{P(A \mid B)P(B)}{P(B)} = \frac{P(B \mid A)P(A)}{P(B)} \rightarrow P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$$

Bayes Rule

Review of basic probability theory

The Bayes rule is a powerful tool we can use to invert the conditional probability. We start off with an **initial estimate of how likely the event A** is when we do not have any other information; **this is the prior probability P(A)**. Bayes' rule lets us 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.

P(A|B) is called the posterior that we are trying to estimate. Example: "probability of a document to be relevant given that its contains a word X".

P(B|A) is called the likelihood; this is the probability of observing the new evidence, given our initial hypothesis. Example: "probability of observing the word X given that the document is relevant".

P(A) is called the prior; this is the probability of our hypothesis without any additional prior information. Example: "probability of any document to be relevant".

P(B) is called the marginal likelihood; this is the total probability of observing the evidence. Example: "probability of observing the word X in any document"

Review of basic probability theory

It is often useful to talk about the **odds of an event**, which provide a kind of multiplier for how probabilities change

$$O(A) = \frac{P(A)}{P(\neg A)} = \frac{P(A)}{1 - P(A)}$$

Probability Ranking Principle

In information retrieval, we represent the relevance of a document given an information need as a random variable R which assumes value 1 when a document is relevant and 0 otherwise. Thus, given a document *d* and a query *q* representing the information need, we want to estimate

$$P(R = 1 \mid q, d)$$

If a reference retrieval system's response to each request is a ranking of the documents in the collection in order of ***decreasing probability of relevance*** *to the user who submitted the request, 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.*

van Rijsbergen, Cornelis Joost. 1979. Information Retrieval, 2nd edition. Butterworths

The binary independence model

Given the dictionary of terms $T = \langle t_1, t_2, \ldots, t_n \rangle$, we model each document as the binary vector $\vec{x} = (x_1, x_2, \ldots, x_n)$, where a component x_i is 1 if the *i*th term of the dictionary is in the document, and 0 otherwise

Using odds and vectors instead of probabilities and documents, we aim at calculating:

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

The binary independence model

Using the Bayes' rule

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

$$= \frac{P(\vec{x} \mid R=1,q)}{P(\vec{x} \mid R=0,q)} \cdot \frac{P(R=1 \mid q)}{P(R=0 \mid q)}$$

The binary independence model

We **assume that terms are independent**. This is a strong assumption because it means for example that the presence of term “Abraham” says nothing about the presence of term “Lincoln”

But, assuming term independence, we can write:

$$O(R=1 \mid q,\vec{x}) = \frac{P(R=1 \mid q)}{P(R=0 \mid q)} \prod_{i=1}^{n} \frac{P(x_i \mid R=1,q)}{P(x_i \mid R=0,q)}$$

This product can be split according to the presence of terms in the document

$$O(R=1 \mid q,\vec{x}) = \frac{P(R=1 \mid q)}{P(R=0 \mid q)} \prod_{i=1}^{n} \frac{P(x_i=1 \mid R=1,q)}{P(x_i=1 \mid R=0,q)} \prod_{i=1}^{n} \frac{P(x_i=0 \mid R=1,q)}{P(x_i=0 \mid R=0,q)}$$

The binary independence model

Now, let's just simplify the formula by introducing

$$p_i = P(x_i = 1 \mid R = 1, q)$$

$$q_i = P(x_i = 1 \mid R = 0, q)$$

	Document	$R = 1$	$R = 0$
We observe the term	$x_i = 1$	p_i	q_i
We do not observe the term	$x_i = 0$	$1 - p_i$	$1 - q_i$

The ranking formula can be simplified as follows, **assuming that $p_1 = q_i$ for all the terms that do not occur in the query**

$$O(R = 1 \mid q, \vec{x}) = \frac{P(R = 1 \mid q)}{P(R = 0 \mid q)} \prod_{x=1, q=1} \frac{p_i}{q_i} \prod_{x=0, q=1} \frac{1 - p_i}{1 - q_i}$$

The binary independence model

Focus only on query terms retrieved in the document

$$O(R=1 \mid q, \vec{x}) = \frac{P(R=1 \mid q)}{P(R=0 \mid q)} \prod_{x=1,q=1} \frac{p_i}{q_i} \prod_{x=0,q=1} \frac{1-p_i}{1-q_i}$$

Now, we can **extend the right product to cover all the query terms** (including those retrieved in the document). But if we want to do so, we have to simultaneously divide the left product by $(1-p_i)/(1-q_i)$ in order to leave the value unchanged

$$O(R=1 \mid q, \vec{x}) = \frac{P(R=1 \mid q)}{P(R=0 \mid q)} \prod_{x_i=1,q_i=1} \frac{\frac{p_i}{q_i}}{\frac{1-p_i}{1-q_i}} \prod_{q_i=1} \frac{1-p_i}{1-q_i} = \frac{P(R=1 \mid q)}{P(R=0 \mid q)} \prod_{x_i=1,q_i=1} \frac{p_i(1-q_i)}{q_i(1-p_i)} \prod_{q_i=1} \frac{1-p_i}{1-q_i}$$

Now we work on all the query terms

The binary independence model

Note that we are interested in evaluating the relative score of a document with respect to the query, not its absolute odds. Thus we observe that:

$$\frac{P(R=1 \mid q)}{P(R=0 \mid q)}$$

This quantity represents the odds of the query and does not depend on the documents

$$\prod_{q_i=1} \frac{1-p_i}{1-q_i}$$

This quantity depends on the terms that are in the query. Thus, it is query dependent but does not change for different documents given the same query

In conclusion, for a given query q, the only document dependent quantity that we need to estimate is

$$O(R=1 \mid q, \vec{x}) = \frac{P(R=1 \mid q)}{P(R=0 \mid q)} \prod_{x_i=1, q_i=1} \frac{p_i(1-q_i)}{q_i(1-p_i)} \prod_{q_i=1} \frac{1-p_i}{1-q_i} \approx \prod_{x_i=1, q_i=1} \frac{p_i(1-q_i)}{q_i(1-p_i)}$$

The binary independence model

According to the previous conclusion, we can calculate the Retrieval Status Value (RSV) for a document given a query as:

$$RSV_{q,d} = \log \prod_{x_i=1,q_i=1} \frac{p_i(1-q_i)}{q_i(1-p_i)} = \sum_{x_i=1,q_i=1} \log \frac{p_i(1-q_i)}{q_i(1-p_i)}$$

Suppose to have some relevance judgments about a set of documents so that R(t) and N(t) are the occurrences of term t in relevant and not relevant docs, respectively

$$p_i = \frac{R(t_i) + 0.5}{R + 1.0} \qquad q_i = \frac{N(t_i) + 0.5}{N + 1.0}$$

Example

R: d1 = [a, b, a, c, d], d2 = [a, d, d, a]

N: d3 = [a, e, f], d4 = [e, f, e, g, d], d5 = [e, f, e, e, g]

Terms	a	b	c	d	e	f	g
R(t)	2	1	1	2	0	0	0
N(t)	1	0	0	1	3	3	2
p_t	0.83	0.5	0.5	0.83	0.17	0.17	0.17
q_t	0.38	0.12	0.12	0.37	0.87	0.87	0.62

New document: d6 = [b, g, e]

$$P(R=1 \mid d6) = \prod_{t \in d6} \frac{p_t(1-q_t)}{q_t(1-p_t)} = \frac{0.5(1-0.12)0.17(1-0.62)0.17(1-0.87)}{0.12(1-0.5)0.62(1-0.17)0.87(1-0.17)} = 0.025$$

The binary independence model

When we do not have information about relevant and not relevant documents, we need to make some **assumptions**:

$$p_i = q_i \ if \ t_i \notin q$$

we restrict product to query terms

$$p = 0.5 \ if \ t_i \in q$$

a query term is likely to be present and absent in a randomly-picked relevant document: p_i and $(1 - p_i)$ cancel out

$$q_i \approx \frac{|\{d \mid t_i \in d\}|}{|D|}$$

non-relevant set of docs is approximated by the whole collection of documents *D* (reasonable because many documents are not relevant)

$$RSV_{q,d} = \sum_{x_i=1,q_i=1} \log \frac{p_i(1-q_i)}{q_i(1-p_i)} = \sum_{x_i=1,q_i=1} \log \frac{(1-q_i)}{q_i} = \sum_{x_i=1,q_i=1} \log \frac{N - N_i + 0.5}{N_i + 0.5}$$

Note that, having n and N as the number of docs with t and the total number of docs, respectively:

$$\log \frac{1-q_i}{q_i} \approx \log \frac{N-n}{n} \approx \log \frac{N}{n} = IDF$$

Non binary models

Probabilistic non binary models take into account the weight of each term in documents instead of just the binary vector

An example is the **Okapi BM25** model, where RSV is estimated as:

$$RSV_{q,d} = \sum_{t \in q} \log \frac{N}{df_t} \frac{(k+1)tf_{t,d}}{k((1-b)+b \times (L_d/L_{avg})) + tf_{td}}$$

Parameter scaling the term frequency

Document length

Average document length

$0 \leq b \leq 1$**: Parameter scaling the document length**

Probabilistic Language Modeling

In order to deal with dependency relations among words in documents, we need a model for:

Compute the probability of a sentence (i.e., a query)

$$P(S) = P(w_1, w_2, \ldots, w_n)$$

Having a generative model for sentences

$$P(w_k \mid w_{k-1}, w_{k-2}, \ldots, w_1)$$

In particular, we aim at **modeling documents** in order to calculate the probability $P(q \mid M_d)$ that a query *q* has been "*generated*" by a document model M_d

Compute a sentence probability

Given a sentence like “gino plays tennis with maria”, we need to compute the joint probability:

`P(gino, plays, tennis, with, maria)`

Using the chain rule

$$P(x_1, x_2, x_3, \ldots, x_n) = P(x_1)P(x_2 \mid x_1)P(x_3 \mid x_1, x_2), \ldots, P(x_n \mid x_1, \ldots, x_{n-1})$$

$$P(w_1, w_2, \ldots, w_n) = \prod_{i}^{n} P(w_i \mid w_1, \ldots, w_{i-1})$$

```
P(gino plays tennis with maria) =
P(gino) x
P(plays | gino) x
P(tennis | gino plays) x
P(with | gino plays tennis) x
P(maria | gino plays tennis with)
```

Compute a sentence probability

Recalling the example, we do not have any way to estimate the probability of "*gino plays tennis with*" or any other complex sentence, because we do not have enough data to observe long terms sequences a sufficient number of times. Thus, we approximate the chain rule as:

$$P(w_1, w_2, \ldots, w_n) \approx \prod_i^n P(w_i \mid w_{i-k}, \ldots, w_{i-1})$$

$$P(w_i \mid w_1, w_2, \ldots, w_{i-1}) \approx \prod_i^n P(w_i \mid w_{i-k}, \ldots, w_{i-1})$$

In practice (examples)

Unigram

$$P(w_1, w_2, \ldots, w_n) \approx \prod_i^n P(w_i)$$

Bi-gram

$$P(w_1, w_2, \ldots, w_n) \approx \prod_i^n P(w_i \mid w_{i-1})$$

Maximum Likelihood estimation

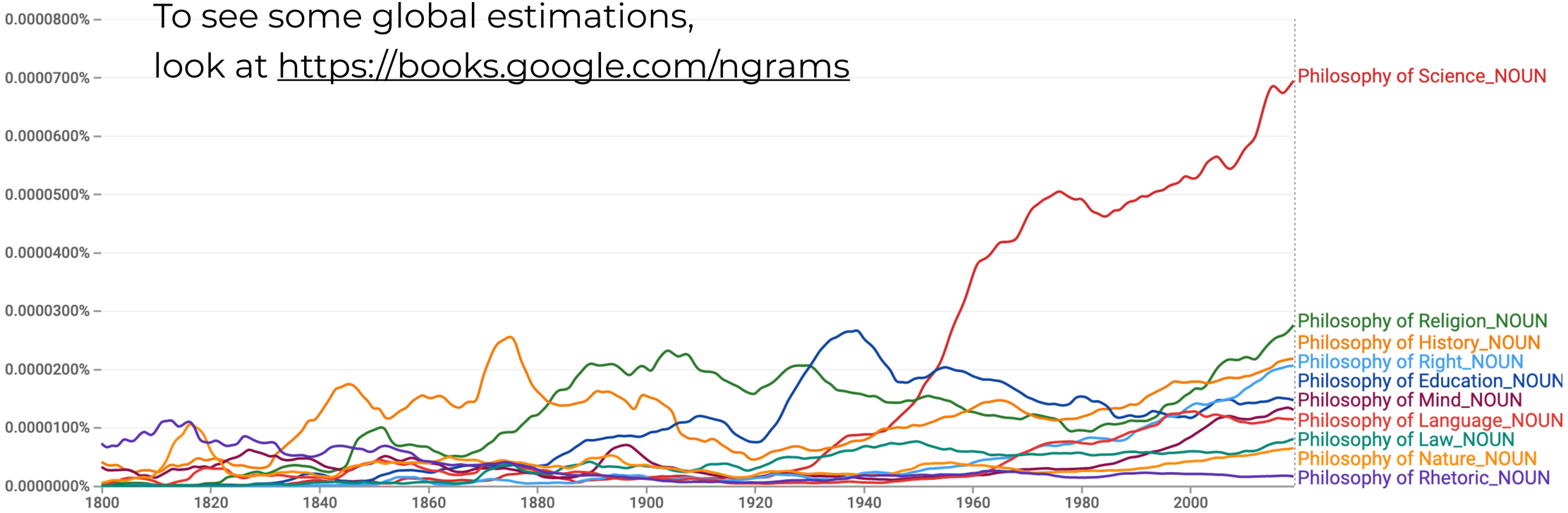
$$P(w_i \mid w_{i-1}) = \frac{count(w_i, w_{i-1})}{\sum_{j=1}^{n} count(w_j, w_{i-1})} = \frac{count(w_i, w_{i-1})}{count(w_{i-1})}$$

Google Books Ngram Viewer

Philosophy of *_NOUN

1800 - 2019 | English (2019) | Case-Insensitive | Smoothing

To see some global estimations,
look at https://books.google.com/ngrams



(click on line/label for focus, right click to expand/contract wildcards)

Query Likelihood model

In the query likelihood model, we generate a model M_d for each document *d* and then we estimate $P(q \mid d)$ as $P(q \mid M_d)$, where

$$P(q \mid M_d) = K_q \prod_i P(w_i \mid M_d)^{tf_{w_i,d}} \qquad \textbf{with } K_q = \frac{Len(K_q)!}{tf_{t1,q}!tf_{t2,q}!\ldots tf_{tN,q}!}$$

K_q is needed to take into account all the equivalent query terms permutations. However, it is strictly query dependent and thus we can just ignore it given the same query over multiple documents

Query Likelihood model

In the query likelihood model, we generate a model M_d for each document *d* and then we estimate $P(q \mid d)$ as $P(q \mid M_d)$, where

$$P(q \mid M_d) = K_q \prod_i P(w_i \mid M_d)^{tf_{w_i,d}} \qquad \textbf{with } K_q = \frac{Len(K_q)!}{tf_{t1,q}!tf_{t2,q}!\ldots tf_{tN,q}!}$$

K_q is needed to take into account all the equivalent query terms permutations. However, it is strictly query dependent and thus we can just ignore it given the same query over multiple documents

Estimating query generation

In the query likelihood model, we generate a model M_d for each document *d* and then we estimate $P(q \mid d)$ as $P(q \mid M_d)$, where

$$P(q \mid M_d) \approx \prod_i \frac{tf_{t_i,d}}{L_d}$$

Estimating query generation

In the query likelihood model, we generate a model M_d for each document *d* and then we estimate $P(q \mid d)$ as $P(q \mid M_d)$, where

$$P(q \mid M_d) \approx \prod_i \frac{tf_{t_i,d}}{L_d}$$

In estimating the probability $P(q \mid M_d)$ all zero and non-frequent term probabilities are a problem, because their role in query evaluation is overestimated

The solution is to **smooth the numbers**, according to different strategies

Smoothing

Laplace smoothing

$$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i) + 1}{count(w_{i-1} + N)}$$

Bayesian smoothing

$$P(w \mid d) = \frac{tf_{t,d} + \alpha P(w \mid M_c)}{L_d + \alpha}$$

Linear Interpolation

where M_c is the model build over the entire document collection

$$P(w \mid d) = \lambda P(w \mid M_d) + (1 - \lambda) P(w \mid M_c)$$

Other models

Kullback-Leibler divergence

$$R(d,q) = \sum_i P(w_i \mid M_q) \log \frac{P(w_i \mid M_q)}{P(w_i \mid M_d)}$$

Translation model

In evaluating the query, we use a generative model in order to take into account terms with high probabilities of being translations for the query terms $P(w \mid t)$

$$P(q \mid M_d) = \prod_i \sum_j^N P(w_j \mid M_d) P(w_i \mid w_j)$$