

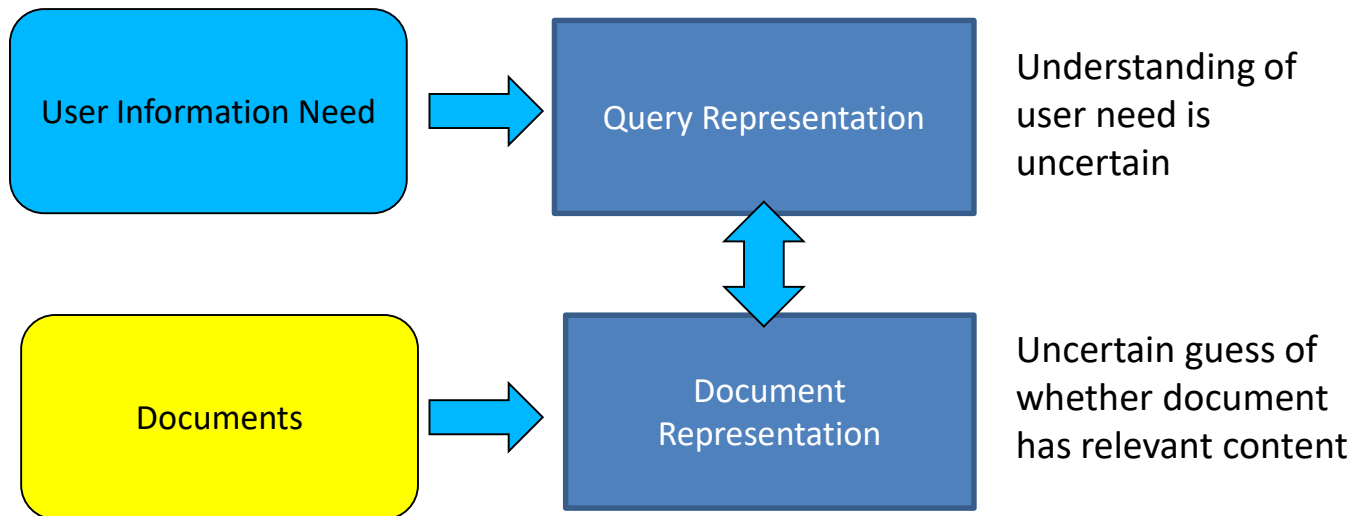
AI6122 Text Data Management & Analysis

Topic: Probabilistic Ranking



Why Probabilistic IR?

- In traditional IR systems, matching between each document and a query is attempted in a **semantically imprecise space** of index terms.
- Can we use probabilities to quantify the uncertainties?



Probabilistic IR topics

- Classical probabilistic retrieval model
- Language model approach to IR



Recall a few probability basics

- Bayes' Rule: for events a and b :

$$P(a, b) = P(a \cap b) = P(a|b)P(b) = P(b|a)P(a)$$

$$P(b) = P(b|a)P(a) + P(b|\bar{a})P(\bar{a})$$

$$P(a|b) = \frac{P(b|a)P(a)}{P(b)} = \frac{P(b|a)P(a)}{\sum_{x=a, \bar{a}} P(b|x)P(x)}$$

- Odds:

$$O(a) = \frac{P(a)}{P(\bar{a})} = \frac{P(a)}{1 - P(a)}$$



Overview

- Probabilistic Approach to Retrieval
- Basic Probability Theory
- Probability Ranking Principle [**For your information only**]
- Language Models for Information Retrieval



The Document Ranking Problem

- Settings: We have a collection of documents
 - User issues a query
 - A list of documents need to be returned
- Ranking method is core of an IR system
 - In what order do we present documents to the user?
 - “Best” documents to be placed top
- Idea: rank by probability of relevance of the document w.r.t. information need
 - $P(\text{Relevance}|\text{document}, \text{query})$



Probability ranking principle (PRP)

- Let x be a document in the collection
- Let R represent **relevance** of a document w.r.t. given query and let NR represent **non-relevance**
- Need to find $P(R|x)$:
 - the probability that a document x is relevant to the query:

$$P(R|x) = \frac{P(x|R)P(R)}{P(x)}$$

$$P(NR|x) = \frac{P(x|NR)P(NR)}{P(x)}$$

- $P(R|x) + P(NR|x) = 1$



Probability ranking principle (PRP)

- Bayes' optimal decision rule: x is relevant iff $P(R|x) > P(NR|x)$
- PRP in action: Rank all documents by $P(R|x)$
- How do we compute all those probabilities?
 - Do not know exact probabilities, have to use **estimates**
 - Simplest model: Binary Independence Retrieval (BIR)
 - Binary means Boolean: document/query is represented by binary term vectors
- (Questionable) Assumptions
 - Relevance of each document is independent from relevance of other documents
 - how about duplicates?
 - Single step information need
 - how about query reformulation?



Binary Independence Model (BIM)

- Traditionally used in conjunction with the PRP
- ‘Binary’ (=Boolean)
 - Documents and queries represented as binary term incidence vectors
 - Document d represented by vector $\vec{x} = (x_1, \dots, x_M)$, where $x_t = 1$ if term t occurs in d and $x_t = 0$ otherwise

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	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



Binary Independence Model (BIM)

- ‘Independence’: no association between terms
 - not true, but practically works
 - ‘naive’ assumption of Naive Bayes models
- Similar to Bernoulli Naïve Bayes model
 - One feature X_w for each word in dictionary
 - $X_w = \text{True}$ in document d if w appears in d
 - Naïve Bayes assumption
 - Given the document’s topic, appearance of one word in the document tells us nothing about the chances that another word appears.



Binary Independence Model (BIM)

- Queries: binary term incidence vectors
- Given query \vec{q} ,
 - for each document d (represented as \vec{x}), compute $P(R|\vec{q}, \vec{x})$
 - Interested only in ranking
- Will use odds and Bayes's Rule:

$$O(R|\vec{q}, \vec{x}) = \frac{P(R|\vec{q}, \vec{x})}{P(NR|\vec{q}, \vec{x})} = \frac{\frac{P(R|\vec{q})P(\vec{x}|R, \vec{q})}{P(\vec{x}|\vec{q})}}{\frac{P(NR|\vec{q})P(\vec{x}|NR, \vec{q})}{P(\vec{x}|\vec{q})}} = \frac{P(R|\vec{q})}{P(NR|\vec{q})} \times \frac{P(\vec{x}|R, \vec{q})}{P(\vec{x}|NR, \vec{q})}$$



Deriving a Ranking Function for Query Terms

SKIP

- Using Independence Assumption:

$$\frac{P(\vec{x}|R, \vec{q})}{P(\vec{x}|NR, \vec{q})} = \prod_i^n \frac{P(x_i|R, \vec{q})}{P(x_i|NR, \vec{q})}$$

- Then we have:

$$O(R|\vec{q}, \vec{x}) = \frac{P(R|\vec{q})}{P(NR|\vec{q})} \times \prod_i^n \frac{P(x_i|R, \vec{q})}{P(x_i|NR, \vec{q})}$$

Document
independent



Deriving a Ranking Function for Query Terms

SKIP

- Since x_i is either 0 or 1: $O(R|\vec{q}, \vec{x}) = O(R|\vec{q}) \cdot \prod_{i=1}^n \frac{P(x_i|R, \vec{q})}{P(x_i|NR, \vec{q})}$

$$O(R|\vec{q}, \vec{x}) = O(R|\vec{q}) \cdot \prod_{x_i=1} \frac{P(x_i = 1|R, \vec{q})}{P(x_i = 1|NR, \vec{q})} \cdot \prod_{x_i=0} \frac{P(x_i = 0|R, \vec{q})}{P(x_i = 0|NR, \vec{q})}$$

- Assume: Ignore all terms not occurring in the query ($q_i=0$)

$$O(R|\vec{q}, \vec{x}) = O(R|\vec{q}) \cdot \prod_{x_i=1, q_i=1} \frac{P(x_i = 1|R, \vec{q})}{P(x_i = 1|NR, \vec{q})} \cdot \prod_{x_i=0, q_i=1} \frac{P(x_i = 0|R, \vec{q})}{P(x_i = 0|NR, \vec{q})}$$

For all matching query
terms

For all non-matching
query terms



Deriving a Ranking Function for Query Terms

SKIP

$$O(R|\vec{q}, \vec{x}) = O(R|\vec{q}) \cdot \prod_{x_i=1, q_i=1} \frac{P(x_i = 1|R, \vec{q})}{P(x_i = 1|NR, \vec{q})} \cdot \prod_{x_i=0, q_i=1} \frac{P(x_i = 0|R, \vec{q})}{P(x_i = 0|NR, \vec{q})}$$

- Let

$$p_i = P(x_i = 1|R, \vec{q})$$
$$r_i = P(x_i = 1|NR, \vec{q})$$

	document	relevant (R)	nonrelevant (NR)
Term present	$x_i = 1$	p_i	r_i
Term absent	$x_i = 0$	$1 - p_i$	$1 - r_i$

P_i is the probability of term x_i appearing in a relevant document

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

Including query terms found in documents into the right product and simultaneously dividing through by them in the left product

Document
independent



Binary Independence Model (BIM)

SKIP

$$O(R|\vec{q}, \vec{x}) = O(R|\vec{q}) \cdot \prod_{x_i=q_i=1} \frac{p_i(1-r_i)}{r_i(1-p_i)} \cdot \prod_{q_i=1} \frac{1-p_i}{1-r_i}$$

Constant for each query

- Retrieval Status Value (RSV)

$$RSV = \log \prod_{x_i=q_i=1} \frac{p_i(1-r_i)}{r_i(1-p_i)} = \sum_{x_i=q_i=1} \log \frac{p_i(1-r_i)}{r_i(1-p_i)} = \sum_{x_i=q_i=1} c_i$$

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



Binary Independence Model (BIM)

[SKIP](#)

- For each term t , look at the table of document counts:

Documents	Relevant	Non-relevant	Total
Term present ($x = 1$)	s	$df_t - s$	df_t
Term absent ($x = 0$)	$S - s$	$(N - df_t) - (S - s)$	$N - df_t$
Total	S	$N - S$	N

$$RSV = \sum_{x_i=q_i=1} c_i \quad \text{where } c_i = \log \frac{p_i(1 - r_i)}{r_i(1 - p_i)}$$

$$\text{Estimates: } p_i \approx \frac{s}{S}, \quad r_i \approx \frac{df_t - s}{N - S}$$

$$c_i \approx \log \frac{\frac{s}{S} \times \left(1 - \frac{df_t - s}{N - S}\right)}{\frac{df_t - s}{N - S} \times \left(1 - \frac{s}{S}\right)} = \log \frac{s/S - s}{df_t - s / N - S - df_t + s}$$



Estimation – key challenge

- r_i (prob. of term occurrence in non-relevant documents for query)
 - If non-relevant documents are approximated by the whole collection, then r_i is $\frac{df_i}{N}$ and
 - $\log \frac{(1-r_i)}{r_i} \approx \log \frac{(N-df_i)}{df_i} \approx \log \frac{N}{df_i} = IDF!$
- p_i (probability of term occurrence in relevant documents) can be estimated in various ways:
 - from relevant documents from user (relevance feedback)
 - constant (e.g. 0.5)
 - proportional to probability of occurrence in collection

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



PRP and BIR

- Among the oldest formal models in IR
 - Since an IR system cannot predict with certainty which document is relevant, we should deal with probabilities
- Requires restrictive assumptions
 - Boolean representation of documents/queries/relevance
 - Term independence
 - Out-of-query terms do not affect retrieval
 - Document relevance values are independent
- Some of the assumptions can be removed
 - A successful method: Okapi BM25 (later)



An Appraisal of Probabilistic Models

- The difference between ‘vector space’ and ‘probabilistic’ IR **is not that great**:
 - In either case you build an information retrieval scheme in the exact same way.
- Difference:
 - 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



Okapi BM25: A Nonbinary Model

- The 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
- From 1994 until today, BM25 is one of the most widely used and robust retrieval models



Okapi BM25: A Nonbinary Model

- The simplest score for document d is just *idf* weighting of the query terms present in the document:
 - $RSV_d = \sum_{t \in q} \log \frac{N}{df_t}$
- Improve this formula by factoring in the term frequency and document length:

$$RSV_d = \sum_{t \in q} \log \left[\frac{N}{df_t} \right] \times \frac{(k_1 + 1) \times tf_{td}}{k_1 \times \left((1 - b) + b \times \left(\frac{L_d}{L_{ave}} \right) \right) + tf_{td}}$$

- 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
- b : tuning parameter controlling the scaling by document length



Okapi BM25: A Nonbinary Model

- If the query is long, we might also use similar weighting for query term

$$RSV_d = \sum_{t \in q} \log \left[\frac{N}{df_t} \right] \times \frac{(k_1 + 1) \times tf_{td}}{k_1 \times \left((1 - b) + b \times \left(\frac{L_d}{L_{ave}} \right) \right) + tf_{td}} \times \frac{(k_3 + 1) \times tf_{tq}}{k_3 + tf_{tq}}$$

- tf_{tq} : term frequency in the query q
- k_3 : tuning parameter controlling term frequency scaling of the query
- The above tuning parameters should ideally be set to optimize performance on a development test collection.
 - In the absence of such optimisation, experiments have shown reasonable values are to set k_1 and k_3 to a value between 1.2 and 2 and $b = 0.75$




Stochastic Language Models

- Model probability of generating strings (each word in turn) in a language
 - commonly all strings over alphabet Σ
 - Example: a unigram model

Model M	
0.20	the
0.10	a
0.01	man
0.01	woman
0.03	said
0.02	likes
...	...

the	man	likes	the	woman
0.2	0.01	0.02	0.2	0.01


$$P(s \mid M) = 0.00000008$$



Stochastic Language Models

- Model probability of generating any string

Model M_1	
0.20	the
0.01	class
0.0001	sayst
0.0001	pleaseth
0.0001	yon
0.0005	maiden
0.01	woman

Model M_2	
0.20	the
0.0001	class
0.03	sayst
0.02	pleaseth
0.1	yon
0.01	maiden
0.0001	woman

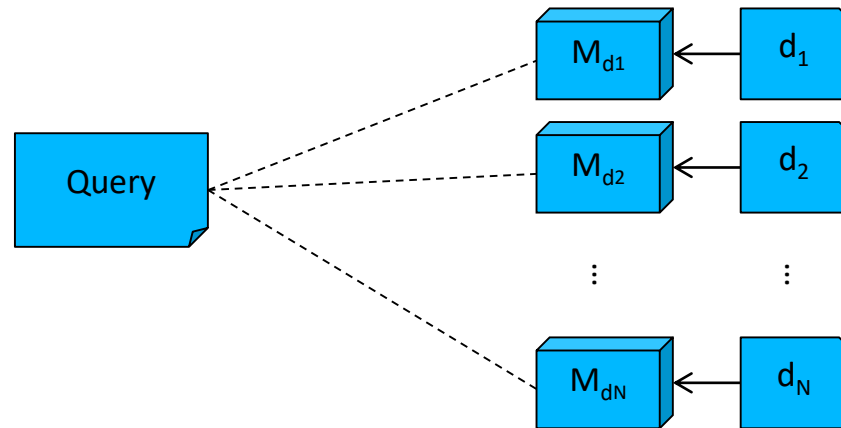
the	class	pleaseth	yon	maiden
0.2	0.01	0.0001	0.0001	0.0005
0.2	0.0001	0.02	0.1	0.01

$$P(s|M_2) > P(s|M_1)$$



Language models (LMs) for IR

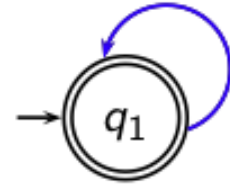
- We view a document as a generative model that generates the query
 - $p(\text{Query}|M_d)$



- How do we create such a model for a document?
- How do we generate a query from language model?

A probabilistic language model

- A language model is a function that puts a probability measure over strings drawn from some vocabulary
 - One-state probabilistic finite-state automaton
 - A unigram language model – and the state emission distribution for its one state q_1
- STOP is not a word, but a special symbol indicating that the automaton stops
 - e.g. $P(\text{"frog said that toad likes frog STOP"})$
 $= 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.2$
 $= 0.00000000000048$



w	$P(w q_1)$	w	$P(w q_1)$
STOP	0.2	toad	0.01
the	0.2	said	0.03
a	0.1	likes	0.02
frog	0.01	that	0.04
	

Using language models in IR

- Each document is treated as (the basis for) a language model.
- Given a query q , rank documents based on $P(d|q) = \frac{P(q|d)P(d)}{P(q)}$
 - $P(q)$ is the same for all documents, so ignore
 - $P(d)$ is the prior – often treated as the same for all d (how about PageRank?)
 - $P(q|d)$ is the probability of q given d .
 - the probability that a query would be observed as a random sample from the respective document model.
 - So to rank documents according to relevance to q , ranking according to $P(q|d)$ and $P(d|q)$ is equivalent.



How to compute $P(q|d)$: query likelihood model

- We will make the same conditional independence assumption as for Naive Bayes.
- $P(q|M_d) = P(< t_1, \dots, t_{|q|} > | M_d) = \prod_{1 \leq k \leq |q|} P(t_k | M_d)$
 - $|q|$: length of q ;
 - t_k : the token occurring at position k in q
 - This is equivalent to the following, where $tf_{t,q}$: term frequency (# occurrences) of t in q
 - $P(q|M_d) = \prod_{\text{distinct term } t \in q} P(t_k | M_d)^{tf_{t,q}}$



Query generation probability

$$p(Q | M_d) = \prod_{t \in Q} p_{mle}(t | M_d) = \prod_{t \in Q} \frac{tf_{t,d}}{dl_d}$$

- Unigram assumption: Words appear independently in document
 - *mle*: Maximum likelihood estimation
 - $tf_{t,d}$: term frequency of term t in document d
 - dl_d : the total number of tokens in document d
 - If a query term does not appear in a document?
 - Need smoothing



Smoothing

- Key intuition: a non-occurring term is possible
 - but no more likely than would be expected by chance in the collection.
- Notation:
$$p_{mle}(t | M_c) = \frac{cf_t}{T}$$
 - M_c : the collection model;
 - cf_t : the number of occurrences of t in the collection (or collection frequency)
 - $T = \sum_t cf_t$: the total number of tokens in the collection.
- We will use $\hat{P}(t|M_c)$ to “smooth” $P(t|d)$ away from zero.



Mixture model

$$p(Q | d) \propto \prod_{t \in Q} \left(\lambda \frac{tf_{t,d}}{dl_d} + (1 - \lambda) \frac{cf_t}{T} \right)$$

- Model: $P(t|d) = \lambda P_{mle}(t|M_d) + (1 - \lambda) P_{mle}(t|M_c)$
- Mixes the probability from the document with the general collection frequency of the word.
 - High value of λ : “conjunctive-like” search – tends to retrieve documents containing all query words.
 - Low value of λ : more disjunctive, suitable for long queries
 - Correctly setting λ is very important for good performance.



LMs vs. Vector Space Model

- The two have common things
 - Both use term frequencies
 - Probabilities are inherently “length-normalized” $[0,1]$
 - Mixing term frequencies in document and collection has an effect similar to idf
 - Terms that are rare in the collection, but frequent in some documents have great influence on the ranking
- Differences
 - LMs: based on probabilistic theory, VSM: linear algebra notion
 - Collection frequency vs. document frequency
 - Other details: term frequency, length normalization

