# Probabilistic IR

Giorgio Gambosi

Corso di Information Retrieval
CdLM in Informatica
Universit di Roma Tor Vergata

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

# Overview

# 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

# Probabilistic IR Models at a Glance

- Classical probabilistic retrieval model
  - Probability ranking principle
    - Binary Independence Model, BestMatch25 (Okapi)
- Bayesian networks for text retrieval
- Language model approach to IR
  - Important recent work, will be covered in the next lecture
- Probabilistic methods are one of the oldest but also one of the currently 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.

# Basic Probability Theory

- 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:

$$P(AB) = P(A \cap B) = P(A|B)P(B) = P(B|A)P(A)$$

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

$$P(\overline{A}B) = P(B|\overline{A})P(\overline{A})$$

- 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:

$$P(B) = P(AB) + P(\overline{A}B)$$

# Basic Probability Theory

Bayes' Rule for inverting conditional probabilities:

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

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:

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

# The Document Ranking Problem

- Ranked retrieval setup: given a collection of documents, the user issues a query, and an ordered list of documents is returned
- Assume binary notion of relevance: $R_{d,q}$ is a random dichotomous variable, such that
  - $R_{d,q} = 1$ if document $d$ is relevant w.r.t query $q$
  - $R_{d,q} = 0$ otherwise
- Probabilistic ranking orders documents decreasingly by their estimated probability of relevance w.r.t. query: $P(R = 1|d, q)$
- Assume that the relevance of each document is independent of the relevance of other documents

# Probability Ranking Principle (PRP)

- 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

# Binary Independence Model (BIM)

- Traditionally used with the PRP

Assumptions:

- 'Binary' (equivalent to Boolean): documents and queries represented as binary term incidence vectors
  - E.g., document $d$ represented by vector $\vec{x} = (x_1, \ldots, x_M)$, where $x_t = 1$ if term $t$ occurs in $d$ and $x_t = 0$ otherwise
  - Different documents may have the same vector representation
- 'Independence': no association between terms (not true, but practically works - 'naive' assumption of Naive Bayes models)

# 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
- Next: how exactly we can do this

# Binary Independence Model

$P(R|d,q)$ is modeled using term incidence vectors as $P(R|\vec{x},\vec{q})$

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

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

- $P(\vec{x}|R=1,\vec{q})$ and $P(\vec{x}|R=0,\vec{q})$: probability that if a relevant or nonrelevant document is retrieved, then that document's representation is $\vec{x}$
- Use statistics about the document collection to estimate these probabilities

# Binary Independence Model

$P(R|d,q)$ is modeled using term incidence vectors as $P(R|\vec{x},\vec{q})$

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

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

- $P(R=1|\vec{q})$ and $P(R=0|\vec{q})$: prior probability of retrieving a relevant or nonrelevant document for a query $\vec{q}$
- Estimate $P(R=1|\vec{q})$ and $P(R=0|\vec{q})$ from percentage of relevant documents in the collection
- Since a document is either relevant or nonrelevant to a query, we must have that:

$$P(R=1|\vec{x},\vec{q}) + P(R=0|\vec{x},\vec{q}) = 1$$

# Deriving a Ranking Function for Query Terms (1)

- Given a query $q$, ranking documents by $P(R = 1|d, q)$ is modeled under BIM as ranking them by $P(R = 1|\vec{x}, \vec{q})$
- Easier: rank documents by their odds of relevance (gives same ranking)

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

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

- $\frac{P(R=1|\vec{q})}{P(R=0|\vec{q})}$ is a constant for a given query - can be ignored

# Deriving a Ranking Function for Query Terms (2)

It is at this point that we make the Naive Bayes conditional independence assumption that the presence or absence of a word in a document is independent of the presence or absence of any other word (given the query):

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

So:

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

# Deriving a Ranking Function for Query Terms (3)

Since each $x_t$ is either 0 or 1, we can separate the terms:

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

# Deriving a Ranking Function for Query Terms (4)

- Let $p_t = P(x_t = 1|R = 1, \vec{q})$ be the probability of a term appearing in relevant document
- Let $u_t = P(x_t = 1|R = 0, \vec{q})$ be the probability of a term appearing in a nonrelevant document
- Can be displayed as contingency table:

| | document | relevant $(R = 1)$ | nonrelevant $(R = 0)$ |
|---|---|---|---|
| Term present | $x_t = 1$ | $p_t$ | $u_t$ |
| Term absent | $x_t = 0$ | $1 - p_t$ | $1 - u_t$ |

# 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

- If $q_t = 0$, then $p_t = u_t$

Now we need only to consider terms in the products that appear in the query:

$$O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t}{u_t} \cdot \prod_{t:x_t=0,q_t=1} \frac{1-p_t}{1-u_t}$$

- 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

Including the query terms found in the document into the right product, but simultaneously dividing by them in the left product, gives:

$$O(R|\vec{x},\vec{q}) = O(R|\vec{q}) \cdot \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} \cdot \prod_{t:q_t=1} \frac{1-p_t}{1-u_t}$$

- The left 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.
- $\rightarrow$ The only quantity that needs to be estimated to rank documents w.r.t a query is the left product
- Hence the Retrieval Status Value (RSV) in this model:

$$RSV_d = \log \prod_{t:x_t=q_t=1} \frac{p_t(1-u_t)}{u_t(1-p_t)} = \sum_{t:x_t=q_t=1} \log \frac{p_t(1-u_t)}{u_t(1-p_t)}$$

# Deriving a Ranking Function for Query Terms

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

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

- The odds ratio is the ratio of two odds: (i) the odds of the term appearing if the document is relevant ($p_t/(1-p_t)$), and (ii) the odds of the term appearing if the document is nonrelevant ($u_t/(1-u_t)$)
- $c_t = 0$: term has equal odds of appearing in relevant and nonrelevant docs
- $c_t$ positive: higher odds to appear in relevant documents
- $c_t$ negative: higher odds to appear in nonrelevant documents

# Term weight $c_t$ in BIM

- $c_t = \log \frac{p_t}{(1-p_t)} - \log \frac{u_t}{1-u_t}$ functions as a term weight.
- Retrieval status value for document $d$: $RSV_d = \sum_{x_t=q_t=1} c_t$.
- 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 (inverted index etc) for the two models.

# How to compute probability estimates

For each term $t$ in a query, estimate $c_t$ in the whole collection using a contingency table of counts of documents in the collection, where $\mathrm{df}_t$ is the number of documents that contain term $t$:

| | documents | relevant | nonrelevant | Total |
|---|---|---|---|---|
| Term present | $x_t = 1$ | $s$ | $\mathrm{df}_t - s$ | $\mathrm{df}_t$ |
| Term absent | $x_t = 0$ | $S - s$ | $(N - \mathrm{df}_t) - (S - s)$ | $N - \mathrm{df}_t$ |
| | Total | $S$ | $N - S$ | $N$ |

$$p_t = s/S$$

$$u_t = (\mathrm{df}_t - s)/(N - S)$$

$$c_t = K(N, \mathrm{df}_t, S, s) = \log \frac{s/(S-s)}{(\mathrm{df}_t - s)/((N - \mathrm{df}_t) - (S - s))}$$

# Avoiding zeros

- If any of the counts is a zero, then the term weight is not well-defined.
- Maximum likelihood estimates do not work for rare events.
- To avoid zeros: add 0.5 to each count (expected likelihood estimation = ELE)
- For example, use $S - s + 0.5$ in formula for $S - s$

# 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

- Assuming that relevant documents are a very small percentage of the collection, approximate statistics for nonrelevant documents by statistics from the whole collection
- Hence, $u_t$ (the probability of term occurrence in nonrelevant documents for a query) is $\mathrm{df}_t/N$ and

$$\log \frac{1-u_t}{u_t} = \log \frac{N-\mathrm{df}_t}{\mathrm{df}_t} \approx \log \frac{N}{\mathrm{df}_t}$$

- This results into

$$c_t = \log \frac{p_t(1-u_t)}{u_t(1-p_t)} \approx \log \frac{p_t}{(1-p_t)} + \log \frac{N}{\mathrm{df}_t}$$

# Probability estimates in adhoc retrieval

- Ad-hoc retrieval: no relevance judgments available
- In this case: assume that $p_t$ is constant over all terms $x_t$ in the query and that $p_t = 0.5$
- Each term is equally likely to occur in a relevant document, and so the $p_t$ and $(1 - p_t)$ factors cancel out in the expression for $RSV$
- Weak estimate, but doesn't disagree violently with expectation that query terms appear in many but not all relevant documents

# Probability estimates in adhoc retrieval

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

$$RSV_d = \sum_{t:x_t=q_t=1} \log \frac{p_t(1-u_t)}{u_t(1-p_t)} \approx \sum_{t:x_t=q_t=1} \log \frac{N}{\mathrm{df}_t}$$

- For short documents (titles or abstracts) in one-pass 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.
- Next: how to add term frequency and length normalization to the probabilistic 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

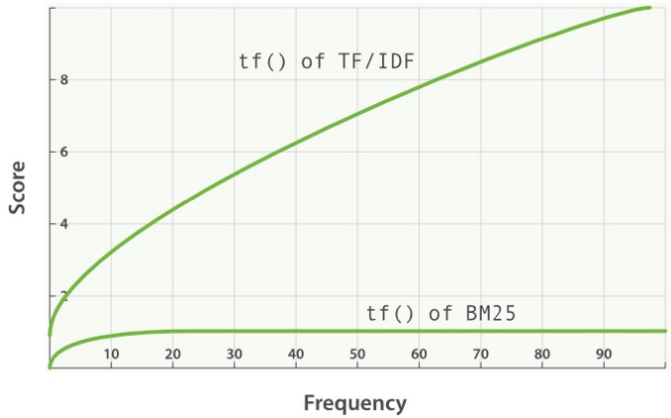
- Improve idf term [log N/df] by factoring in term frequency.

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

- $k_1$: 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 $\mathrm{tf}_{td}$ values
- Why might documents be longer?
  - Verbosity: suggests observed $\mathrm{tf}_{td}$ too high
  - Larger scope: suggests observed $\mathrm{tf}_{td}$ 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
$$L_d = \sum_t \mathrm{tf}_{td}$$
- $L_{\text{ave}}$: average document length
- 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 and is s multiplied by the ratio of the field length we just discussed.
- If $b$ is bigger, the effects of the length of the document compared to the average length are more amplified.
- Usually, $b$ has a value around 0.75.

# Okapi BM25 basic weighting

- Improve idf term [log N/df] by factoring in term frequency and document length.

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

- $\mathrm{tf}_{td}$: term frequency in document $d$
- $L_d$ ($L_{\mathrm{ave}}$): 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

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

- $\mathrm{tf}_{tq}$: 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 $\rightarrow$ use vector space with tf-idf weighting.
- I want to use a state-of-the-art ranking model with excellent performance $\rightarrow$ use BM25 (or language models) with tuned parameters
- In between: BM25 or language models with no or just one tuned parameter