CS3245

Information Retrieval

11

Lecture 11: Probabilistic IR and Language Models for IR

Last Time

- XML Retrieval
  - Lexicalized Subtrees
    - *A la* zones, fields
  - Context Resemblance

- XML Evaluation
  - Content and Structure
  - Partial Relevance

- Relevance Feedback
  - Documents
- Query Expansion
  - Terms

1. Explicit (Uncommon)
2. Implicit (Common for Web)
3. Blind / Pseudo RF

Today

Chapter 11

1. Probabilistic Approach to Retrieval / Basic Probability Theory
2. Probability Ranking Principle
3. OKAPI BM25

Chapter 12

1. Language Models for IR

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
- Boolean or vector space models of IR: query-document matching done in a formally defined but semantically imprecise calculus of index terms
    - 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)

- Language model approach to IR
  - Important recent work, competitive performance

Probabilistic methods are one of the oldest but also one of the currently hottest topics in IR

Basic Probability Theory

- For events A and B
  - Joint probability P(*A*, *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:

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

- 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:

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

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 multiplier for how probabilities change:

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

THE PROBABILITY RANKING PRINCIPLE

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 binary variable, such that:
  - $R_{d,q} = 1$ if document *d* is relevant to query *q*
  - $R_{d,q} = 0$ otherwise
- **Probabilistic ranking** orders documents decreasingly by their estimated probability of relevance to the query: $P(R = 1|d, q)$

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 $\vec{x} = (x_1, . . . , x_M)$, 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 works in practice – **naïve** assumption)

Binary Independence Model

- To make a probabilistic retrieval strategy precise, we need to estimate how terms in documents contribute to relevance
  - Find measurable statistics that affect judgments about document relevance
  - Use them to estimate the probability of relevance $P(R|d, q)$
  - Order documents by decreasing probability values.
  - Assume: Relevance of documents are independent (not true; duplicate results considered bad)

Binary Independence Model

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

$$
\begin{aligned}
P(R=1|\vec{x},\vec{q}) &= \frac{P(\vec{x}|R=1,\vec{q})P(R=1|\vec{q})}{P(\vec{x}|\vec{q})} \\
P(R=0|\vec{x},\vec{q}) &= \frac{P(\vec{x}|R=0,\vec{q})P(R=0|\vec{q})}{P(\vec{x}|\vec{q})}
\end{aligned}
$$

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

Binary Independence Model

Same equation as on previous slide

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

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

- $P(R=1|\vec{q})$ and $P(R=0|\vec{q})$: prior probability of retrieving a relevant or nonrelevant document for a query *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 have:

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

Deriving a Ranking Function for Query Terms

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

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

$$\boxed{\text{Drop this term} \quad = \frac{P(R=1|\vec{q})}{P(R=0|\vec{q})}} \cdot \frac{P(\vec{x}|R=1,\vec{q})}{P(\vec{x}|R=0,\vec{q})}$$

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

Deriving a Ranking Function for Query Terms

- It is at this point that we make the (Naïve 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:

$$O(R|\vec{x},\vec{q}) = \underbrace{\boxed{O(R|\vec{q})}}_{\text{Dropped term}} \cdot \underbrace{\boxed{\prod_{t=1}^{M} \frac{P(x_t|R=1,\vec{q})}{P(x_t|R=0,\vec{q})}}}_{\text{Just focus on this term}}$$

Deriving a Ranking Function for Query Terms

- Since each x_t is either present (1) or absent (0), we can separate the terms to give:

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

  - 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
- Visualise this as a 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. I.e.,

  $\boxed{\text{If } q_t = 0\text{, then } p_t = u_t}$

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

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

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

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

Deriving a Ranking Function for Query Terms

- So it boils down to computing RSV. We can rank documents using the log odds ratios for the terms in the query c_t :

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

  - The odds ratio is the ratio of two odds:
    (i) the odds of the term appearing if relevant ($p_t/(1 - p_t)$), and
    (ii) the odds of the term appearing if nonrelevant ($u_t/(1 - u_t)$)
  - c_t = 0 if a term has equal odds of appearing in relevant and nonrelevant documents, and c_t is positive if it is more likely to appear in relevant documents.
  - c_t functions as a term weight, so that $RSV_d = \sum_{x_t=q_t=1} c_t$.
- Operationally, we sum c_t quantities in accumulators for query terms appearing in documents, just like in the VSM.

Probability Estimates in **Theory**

- 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 df_t is the number of documents that contain term *t*:

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

$$p_t = s/S$$ ← Term present in relevant document

$$u_t = (\mathrm{df}_t - s)/(N - S)$$ ← Term absent in relevant document

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

- To avoid zero probabilities (such as if every or no relevant document has a particular term), we apply **smoothing**.

Probability Estimates in Practice

- Assuming that relevant documents are a very small percentage of the collection, we 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 df_t/N and,

$$\log[(1 - u_t)/u_t] = \log[(N - \mathrm{df}_t)/\mathrm{df}_t] \approx \log N/\mathrm{df}_t$$

- But note that the above approximation cannot easily be extended to relevant documents.

Probability Estimates in Practice

- Statistics of relevant documents (p_t) can be estimated in various ways:
  1. Use the frequency of term occurrence in known relevant documents (if known).
  2. Set as constant. E.g., 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 pt 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
     - 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
     - For short documents (titles or abstracts) in one-pass retrieval situations, this estimate can be quite satisfactory

AN APPRAISAL OF PROBABILISTIC MODELS

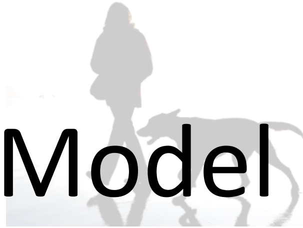
An Appraisal of Probabilistic Models

- Among the oldest formal models in IR
  - (Maron and Kuhns, 1960) Since an IR system cannot predict with certainty which document is relevant, we should deal with probabilities

- Assumptions for getting reasonable approximations of the needed probabilities (in the BIM):
  - Boolean representation of documents/queries/relevance
  - Term independence
  - Out-of-query terms do not affect retrieval
  - Document relevance values are independent

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 (i.e., 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

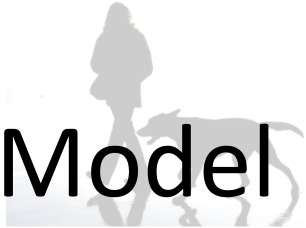
- The simplest score for document *d* is just *idf* weighting of the query terms present in the document:

$$RSV_d = \sum_{t \in q} \log \frac{N}{\mathrm{df}_t}$$

- Improve this formula by factoring in the term frequency and document length:

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

- 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

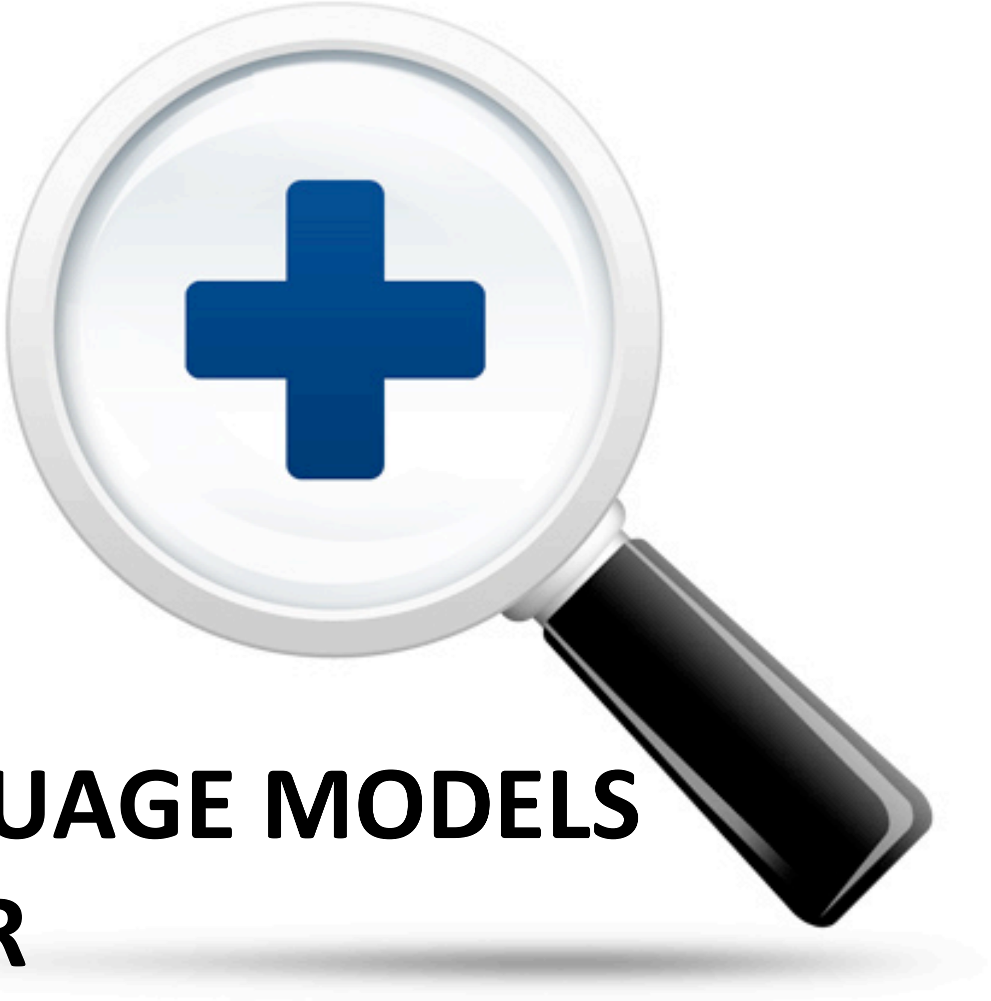


Okapi BM25: A Nonbinary Model

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

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

- 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 be set by optimization on a development test collection. Experiments have shown reasonable values for k_1 and k_3 as values between 1.2 and 2 and b = 0.75



LANGUAGE MODELS FOR IR

LM generative models

- We want to classify a query *q*.
  - Each document in the collection is a different class.
- Assume that *q* was generated by a generative model.
- **Key question**: Which document (= class) is most likely to have generated the query *q*?
  - Or: For which document (as the source of the query) do we have the most evidence?

Using language models (LMs) for IR

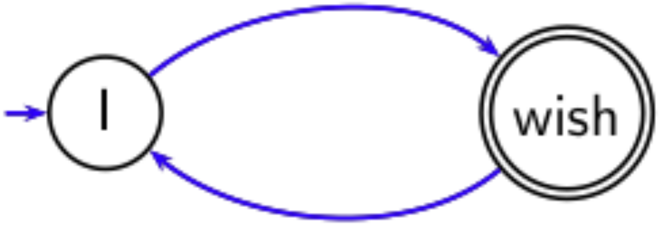
- View the document as a model that generates the query

What we need to do:

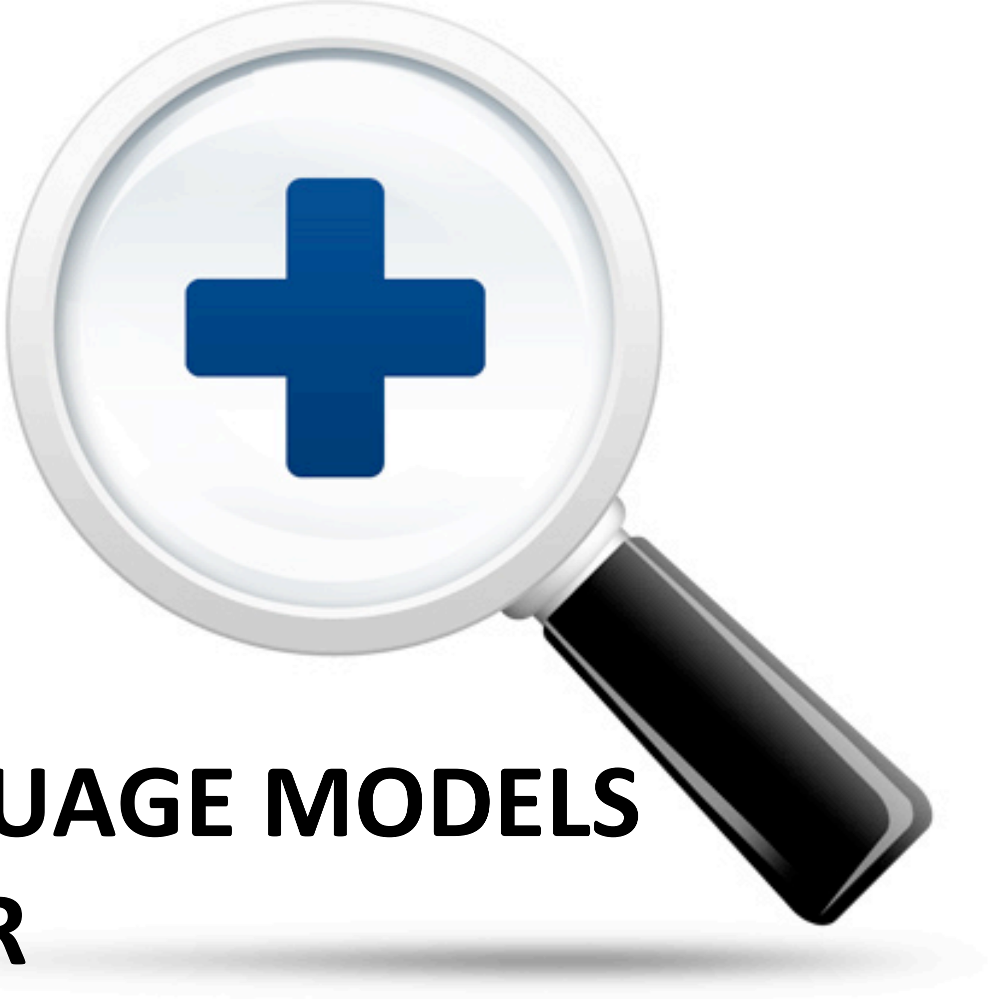
1. Define the precise generative model we want to use
2. Estimate parameters (different parameters for each document's model)
3. Smooth to avoid zeros
4. Apply to query and find document most likely to have generated the query
5. Present most likely document(s) to user

Recap: What is a language model?

- We can view a finite state automaton as a deterministic language model.



- I wish I wish I wish I wish . . . Cannot generate: “wish I wish” or “I wish I”
- Our basic model: each document was generated by a different automaton like this except that these automata are probabilistic



LANGUAGE MODELS FOR IR

Using language models in IR

- Each document is treated as (the basis for) a language model.
- Given a query *q, r*ank documents based on *P*(*d*|*q*)

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)}$$

- *P*(*q*) is the same for all documents, so ignore
- *P*(*d*) is the prior – often treated as the same for all *d*
  - But we can give a prior to "high-quality" documents, e.g., those with high static quality score g(d) (cf. Section 7.14).
- *P*(*q*|*d*) is the probability of *q* given *d.*
- So to rank documents according to relevance to *q*, ranking according to *P*(*q*|*d*) and *P*(*d*|*q*) is equivalent.

Where we are

- In the LM approach to IR, we attempt to model the query generation process.
- Then we rank documents by the probability that a query would be observed as a random sample from the respective document model.
- That is, we rank according to $P(q|d)$.
- Next: how do we compute $P(q|d)$?

How to compute $P(q|d)$

- We make a conditional independence assumption.

$$P(q|M_d) = P(\langle t_1, \ldots, t_{|q|}\rangle|M_d) = \prod_{1\leq k\leq|q|} P(t_k|M_d)$$

(|*q*|: length of *q*; t_k : the token occurring at position *k* in *q*)

- This is equivalent to:

$$P(q|M_d) = \prod_{\text{distinct term } t \text{ in } q} P(t|M_d)^{\mathrm{tf}_{t,q}}$$

$tf_{t,q}$: term frequency (# occurrences) of *t* in *q*

- Multinomial model (omitting constant factor)

Parameter Estimation

- Missing piece: Where do the parameters $P(t|M_d)$. come from?
- Start with maximum likelihood estimates:

$$\hat{P}(t|M_d) = \frac{\mathrm{tf}_{t,d}}{|d|}$$

  (|*d*|: length of *d*; $\mathrm{tf}_{t,d}$: # occurrences of *t* in *d*)
- But a single *t* with $P(t|M_d) = 0$ will make $P(q|M_d) = \prod P(t|M_d)$ zero. Essentially, we have given a single term "veto power".
- For example, for query [Michael Jackson top hits] a document about "top songs" (not using the word "hits") would have $P(t|M_d) = 0$. That's bad.
- We need to smooth the estimates to avoid zeros.

Smoothing, revisited

- Key intuition: A non-occurring term is possible (even though it didn't occur), . . .

  . . . but no more likely than would be expected by chance in the collection.

- Notation: M_c: the collection model; cf_t: the number of occurrences of *t* in the collection; $T = \sum_t \mathrm{cf}_t$: the total number of tokens in the collection.

$$\hat{P}(t|M_d) = \frac{\mathrm{tf}_{t,d}}{|d|}$$

- We will use $\hat{P}(t|M_c)$ to "smooth" $P(t|d)$ away from zero.

Mixture model

$$P(t|d) = \lambda P(t|M_d) + (1 - \lambda)P(t|M_c)$$

- 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.

Mixture model: Summary

$$P(q|d) \propto \prod_{1 \leq k \leq |q|} (\lambda P(t_k|M_d) + (1-\lambda)P(t_k|M_c))$$

- What we model: The user has a document in mind and generates the query from this document.
- The equation represents the probability that the document that the user had in mind was in fact this one.

Blanks on slides, you may want to fill in

Exercise 1

- Collection: d_1 and d_2
- d_1 : Jackson was one of the most talented entertainers of all time
- d_2: Michael Jackson anointed himself King of Pop
- Query q: Michael Jackson

Use mixture model with $\lambda = 1/2$

- $P(q|d_1) = [(0/11 + 1/18)/2] \cdot [(1/11 + 2/18)/2]$
- $P(q|d_2) = [(1/7 + 1/18)/2] \cdot [(1/7 + 2/18)/2]$
- Ranking:

Blanks on slides, you may want to fill in

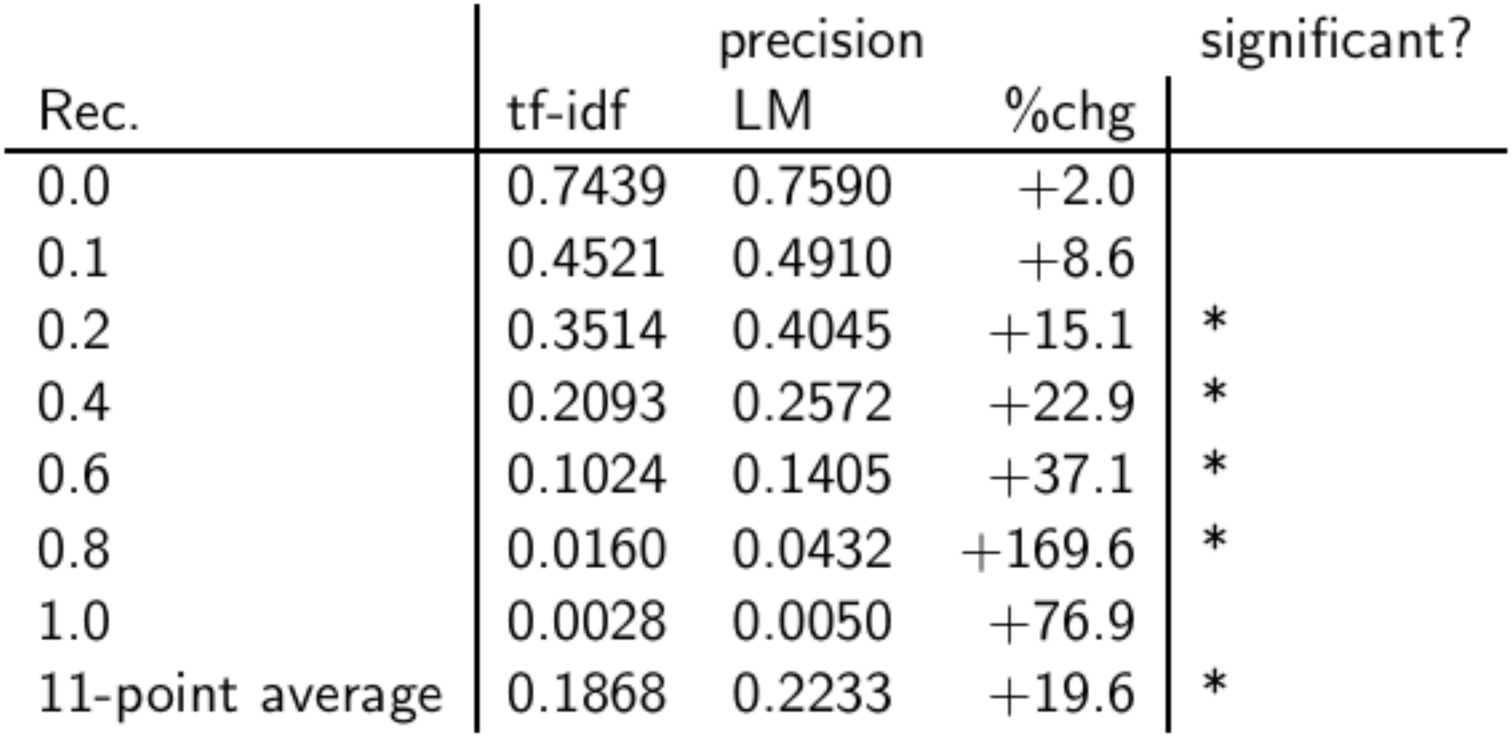
Exercise 2

- Collection: d_1 and d_2
- d_1 : Xerox reports a profit but revenue is down
- d_2: Lucent narrows quarter loss but decreases further
- Query q: revenue down

Use mixture model with λ = ½

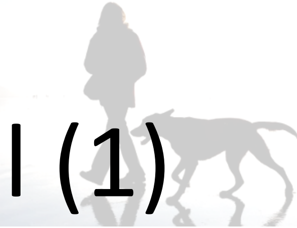
- $P(q|d_1) = [(1/8 + 1/15)/2] \cdot [(1/8 + 1/15)/2] =$
- $P(q|d_2) = [(0/7 + 1/15)/2] \cdot [(0/7 + 1/15)/2] =$
- Ranking:

Vector space (tf-idf) vs. LM



	precision			significant?
Rec.	tf-idf	LM	%chg	
0.0	0.7439	0.7590	+2.0	
0.1	0.4521	0.4910	+8.6	
0.2	0.3514	0.4045	+15.1	*
0.4	0.2093	0.2572	+22.9	*
0.6	0.1024	0.1405	+37.1	*
0.8	0.0160	0.0432	+169.6	*
1.0	0.0028	0.0050	+76.9	
11-point average	0.1868	0.2233	+19.6	*

- The language modeling approach always does better in these experiments . . . but note that where the approach shows significant gains is at higher levels of recall.



LMs vs. vector space model (1)

LMs have some things in common with vector space models.

- Term frequency is directly in the model.
    - But it is not scaled in LMs.
- Probabilities are inherently "length-normalized".
    - Cosine normalization does something similar for vector space.
- Mixing document and collection frequencies has an effect similar to idf.
    - Terms rare in the general collection, but common in some documents will have a greater influence on the ranking.

LMs vs. vector space model (2)

Differences

- LMs: based on probability theory
- Vector space: based on similarity, a geometric/ linear algebra notion
- Collection frequency vs. document frequency
- Details of term frequency, length normalization etc.

Language models for IR: Assumptions

- Simplifying assumption: Queries and documents are objects of same type. Not true!
  - There are other LMs for IR that do not make this assumption.
  - The vector space model makes the same assumption.
- Simplifying assumption: Terms are conditionally independent.
  - VSM also makes the same assumption.
- Cleaner statement of assumptions than vector space
- Thus, better theoretical foundation than vector space
  - … but "pure" LMs perform much worse than "tuned" LMs.

Summary

- Probabilistically grounded approach to IR
  - Probability Ranking Principle
  - Models: BIM, OKAPI BM25
- Language Models for IR

Resources:

- Chapters 11 and 12 of IIR
- Ponte and Croft's 1998 SIGIR paper (one of the first on LMs in IR)
- Lemur toolkit (good support for LMs in IR, http://www.lemurproject.org/)