BLG 540E
TEXT RETRIEVAL SYSTEMS

Probabilistic IR and Language Modeling

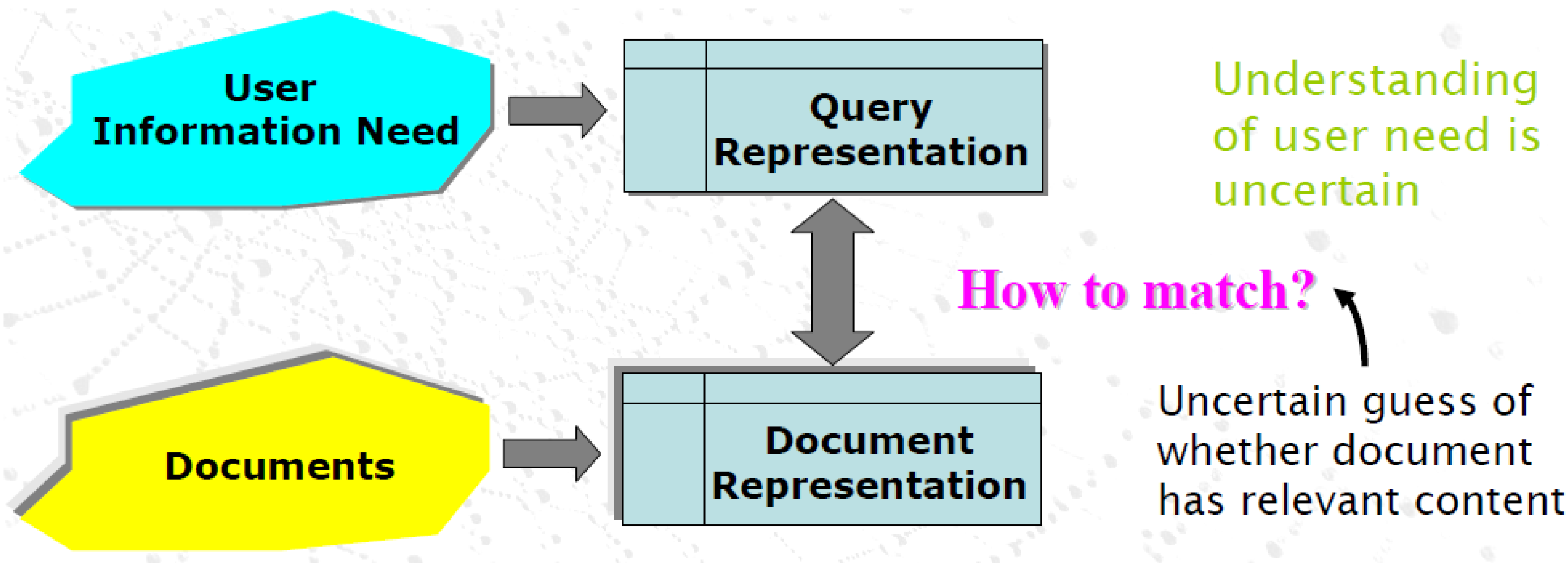
Arzucan Özgür

Faculty of Computer and Informatics, İstanbul Techical University

March 25, 2011

Probabilistic IR

Probabilistic Approach to Retrieval



- 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, 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:

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

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**

Probability Ranking Principle

Let *x* be a document in the collection.
Let *R* represent **relevance** of a document w.r.t. given (fixed) query and let *NR* represent **non-relevance.**

R={0,1} vs. NR/R

Need to find p(*R*|*x*) - probability that a document *x* is **relevant.**

$$p(R\,|\,x) = \frac{p(x\,|\,R)\,p(R)}{p(x)}$$

$$p(NR\,|\,x) = \frac{p(x\,|\,NR)\,p(NR)}{p(x)}$$

p(*R*),p(*NR*) - prior probability of retrieving a (non) relevant document

$$p(R\,|\,x) + p(NR\,|\,x) = 1$$

p(*x*|*R*), p(*x*|*NR*) - probability that if a relevant (non-relevant) document is retrieved, it is *x*.

Binary Independence Model

- Traditionally used in conjunction with PRP
- **"Binary" = Boolean**: documents are represented as binary incidence vectors of terms:
  - $\vec{x} = (x_1, \ldots, x_n)$
  - $x_i = 1$ <u>iff</u> term *i* is present in document *x*.
- **"Independence":** terms occur in documents independently
- Different documents can be modeled as same vector

Binary Independence Model

- Queries: binary term incidence vectors
- Given query ***q***,
  - for each document ***d*** need to compute ***p(R|q,d)***.
  - replace with computing ***p(R|q,x)*** where ***x*** is binary term incidence vector representing ***d***
  - Interested only in ranking
- Will use odds and Bayes' Rule:

$$O(R \mid q, \vec{x}) = \frac{p(R \mid q, \vec{x})}{p(NR \mid q, \vec{x})} = \frac{\dfrac{p(R \mid q)\, p(\vec{x} \mid R, q)}{p(\vec{x} \mid q)}}{\dfrac{p(NR \mid q)\, p(\vec{x} \mid NR, q)}{p(\vec{x} \mid q)}}$$

Binary Independence Model

$$O(R \mid q, \vec{x}) = \frac{p(R \mid q, \vec{x})}{p(NR \mid q, \vec{x})} = \frac{p(R \mid q)}{p(NR \mid q)} \cdot \frac{p(\vec{x} \mid R, q)}{p(\vec{x} \mid NR, q)}$$

Constant for a given query

Needs estimation

- Using **Independence** Assumption:

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

- So : $O(R \mid q, d) = O(R \mid q) \cdot \prod_{i=1}^{n} \frac{p(x_i \mid R, q)}{p(x_i \mid NR, q)}$

Binary Independence Model

$$O(R \mid q, d) = O(R \mid q) \cdot \prod_{i=1}^{n} \frac{p(x_i \mid R, q)}{p(x_i \mid NR, q)}$$

- Since x_i is either 0 or 1:

$$O(R \mid q, d) = O(R \mid q) \cdot \prod_{x_i=1} \frac{p(x_i = 1 \mid R, q)}{p(x_i = 1 \mid NR, q)} \cdot \prod_{x_i=0} \frac{p(x_i = 0 \mid R, q)}{p(x_i = 0 \mid NR, q)}$$

- Let $p_i = p(x_i = 1 \mid R, q); \quad r_i = p(x_i = 1 \mid NR, q);$

  - Assume, for all terms not occurring in the query ($q_i=0$) $p_i = r_i$

Then...

Binary Independence Model

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

All matching terms

Non-matching query terms

$$= O(R \mid q) \cdot \prod_{x_i = q_i = 1} \frac{p_i (1 - r_i)}{r_i (1 - p_i)} \cdot \prod_{q_i = 1} \frac{1 - p_i}{1 - r_i}$$

All matching terms

All query terms

Binary Independence Model

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

Constant for each query

Only quantity to be estimated for rankings

- Retrieval Status Value:

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

Binary Independence Model

- All boils down to computing RSV.

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

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

So, how do we compute c_i's from our data ?

Binary Independence Model

- Estimating RSV coefficients.
- For each term *i* look at this table of document counts:

Documens	Relevant	Non-Relevant	Total
$X_i=1$	s	n-s	n
$X_i=0$	S-s	N-n-S+s	N-n
Total	S	N-S	N

- Estimates: $p_i \approx \frac{s}{S}$ $\quad r_i \approx \frac{(n-s)}{(N-S)}$

$$c_i \approx K(N,n,S,s) = \log \frac{s/(S-s)}{(n-s)/(N-n-S+s)}$$

Probability Estimates in Practice

- Assuming that relevant documents are a very small percentage of the collection, approximate statistics for nonrelevant documents by statistics from the whole collection
- Hence, r_t (the probability of term occurrence in nonrelevant documents for a query) is df_t / N and

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

- The above approximation cannot easily be extended to relevant documents

Prabability 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 pt 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 r_t, the document ranking is determined simply by which query terms occur in documents scaled by their idf weighting

An Appraisal of Probabilistic Models

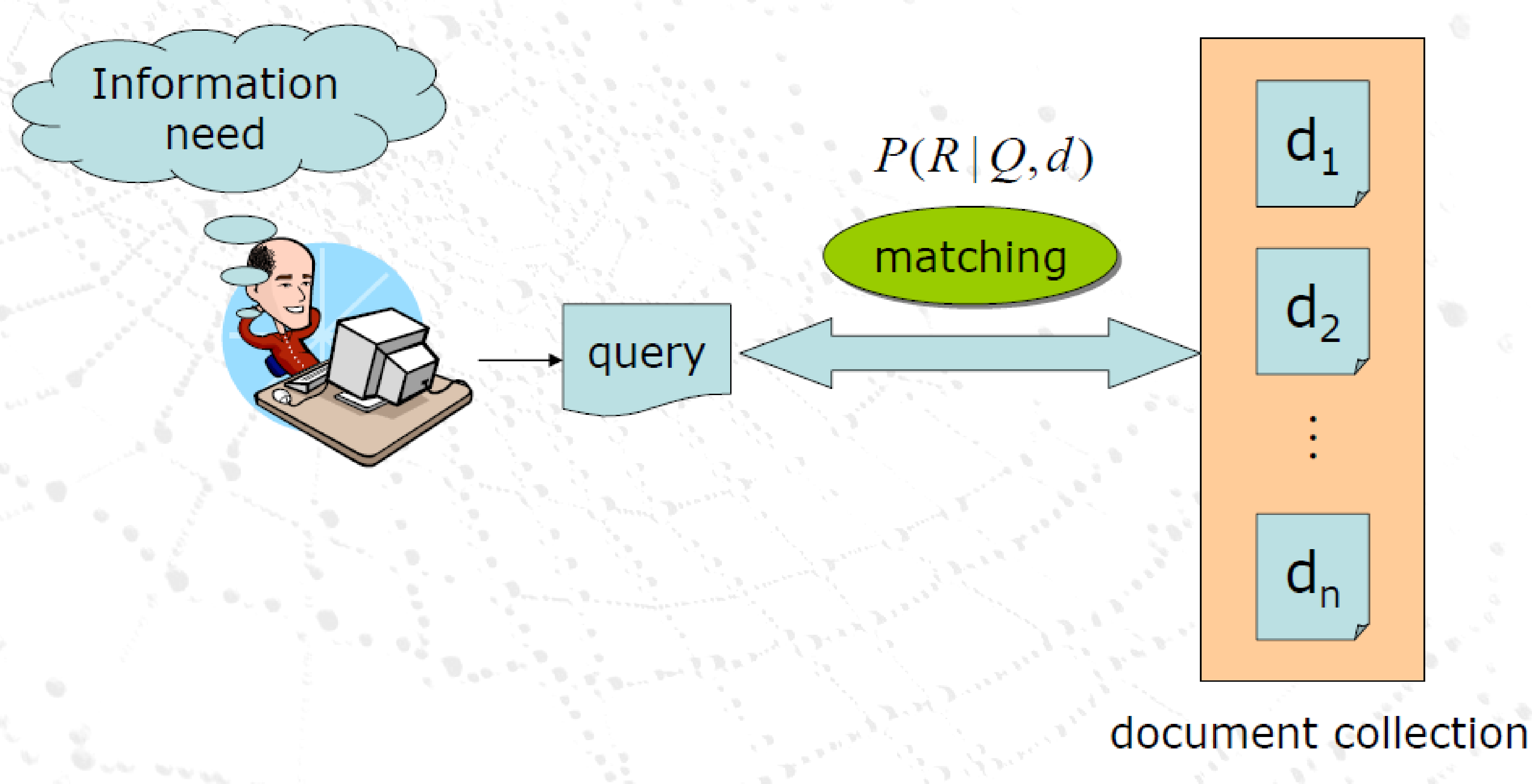
- Among the oldest formal models in IR
  - Maron & Kuhns, 1960: Since an IR system cannot predict with certainty which document is relevant, we should deal with probabilities
- Assumptions for getting reasonable approximations of the needed probabilities (in the BIM):
  - Boolean representation of documents/queries/relevance
  - Term independence
  - Out-of-query terms do not affect retrieval
  - Document relevance values are independent

An Appraisal of Probabilistic Models

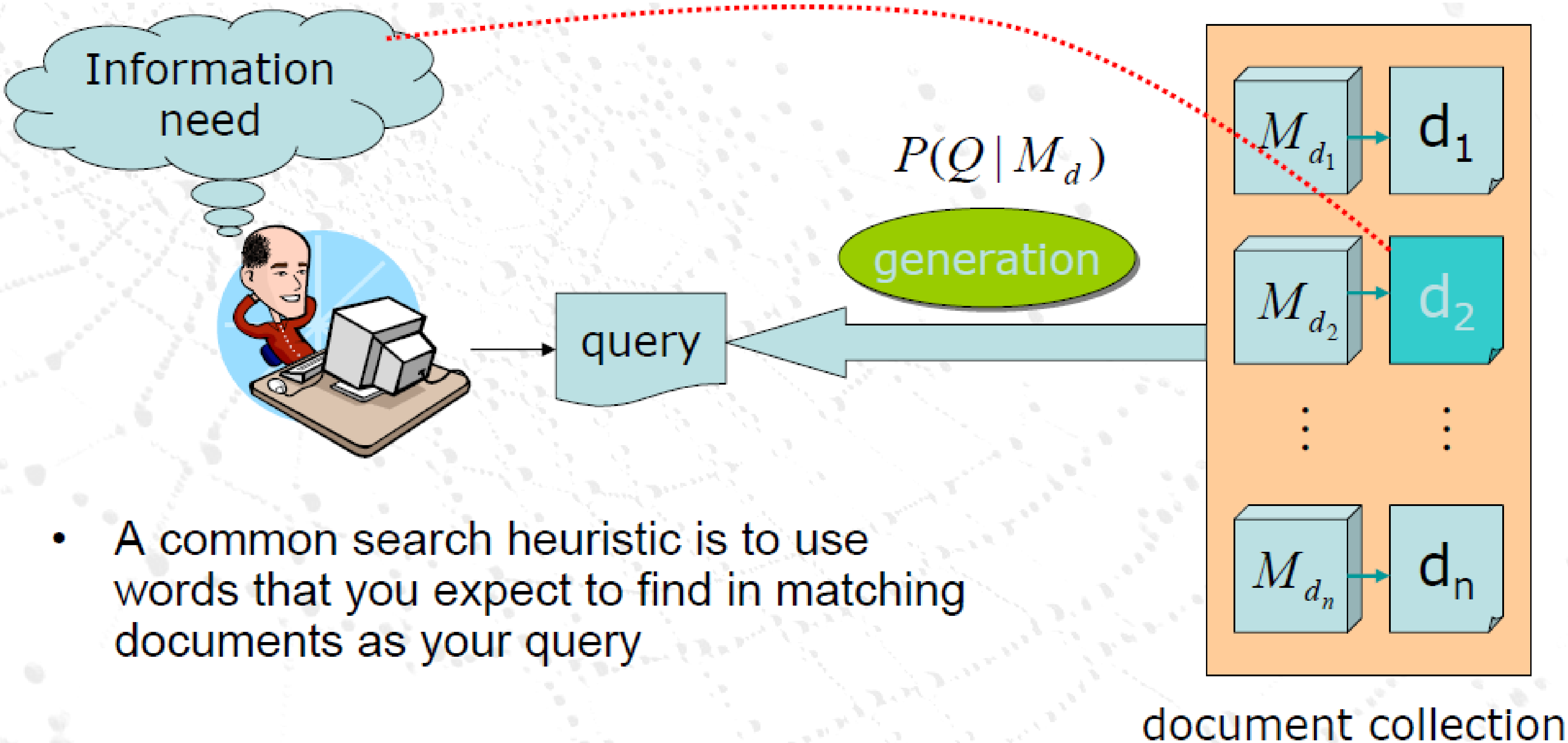
- The difference between 'vector space' and 'probabilistic' IR is not that great:
  - In either case you build an information retrieval scheme in the exact same way.
  - Difference: for probabilistic IR, at the end, you score queries not by cosine similarity and tf-idf in a vector space, but by a slightly different formula motivated by probability theory

Language Models for IR

Standard Probabilistic IR



Language Modeling based IR

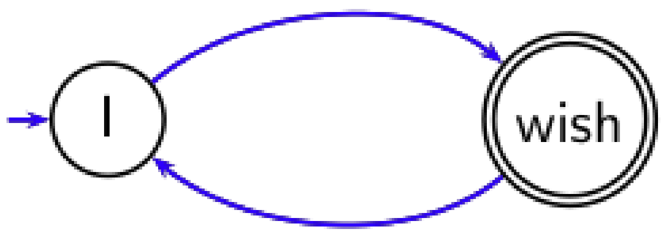


- A common search heuristic is to use words that you expect to find in matching documents as your query

The LM approach directly exploits that idea!

What is a language model?

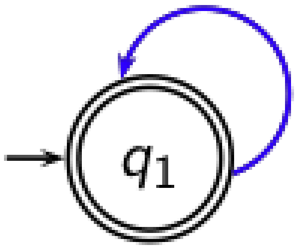
We can view a finite state automaton as a deterministic language model



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.

A probabilistic language model



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
		...	...

This is a 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.

frog said that toad likes frog STOP

$P(\text{string}) = 0.01 \cdot 0.03 \cdot 0.04 \cdot 0.01 \cdot 0.02 \cdot 0.01 \cdot 0.02$

$= 0.0000000000048 = 4.8 \cdot 10^{-12}$

A different language model for each document

language model of d_1

w	$P(w\|.)$	w	$P(w\|.)$
STOP	.2	toad	.01
the	.2	said	.03
a	.1	likes	.02
frog	.01	that	.04
		...	...

language model of d_2

w	$P(w\|.)$	w	$P(w\|.)$
STOP	.2	toad	.02
the	.15	said	.03
a	.08	likes	.02
frog	.01	that	.05
		...	...

$P(\text{string}|M_{d2}) = 0.01 \cdot 0.03 \cdot 0.05 \cdot 0.02 \cdot 0.02 \cdot 0.01 \cdot 0.02 = 0.0000000000120 = 12 \cdot 10^{-12}$

$P(\text{string}|M_{d1}) < P(\text{string}|M_{d2})$

Thus, document d_2 is "more relevant" to the string "frog said that toad likes frog STOP" than d_1 is.

Unigram and Higher Order Models

P (● ● ● ●)

= P (●) P (● | ●) P (● | ● ●) P (● | ● ● ●)

- Unigram Language Models

  P (●) P (●) P (●) P (●)

Easy.
Effective!

- Bigram (generally, *n*-gram) Language Models

  P (●) P (● | ●) P (● | ●) P (● | ●)

- Other Language Models
  - Probably too complex for current IR

Using language models in IR

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

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

- $P(q)$ 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 PageRank.
- $P(q|d)$ 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.

How to compute $P(q|d)$

- We will make the same conditional independence assumption as for Naive Bayes.

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

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

- This is equivalent to:

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

- $\mathrm{tf}_{t,q}$: term frequency (# occurrences) of t in q

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*)

- Problem with zeros.
- A single t with $P(t|M_d) = 0$ will make $P(q|M_d) = \prod P(t|M_d)$ zero.
- For example, for query [Michael Jackson top hits] a document about "top songs" (but 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

- Key intuition: A nonoccurring 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.

Example

- 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 $\lambda = 1/2$
- $P(q|d_1) = [(0/11 + 1/18)/2] \cdot [(1/11 + 2/18)/2] \approx 0.003$
- $P(q|d_2) = [(1/7 + 1/18)/2] \cdot [(1/7 + 2/18)/2] \approx 0.013$
- Ranking: $d_2 > d_1$

Exercise:

- Collection: d_1 and d_2
- d_1 : Xerox reports a profit but revenue is down
- d_2: Lucene narrows quarter loss but decreases further
- Query q: revenue down
- Use mixture model with $\lambda = 1/2$
- $P(q|d_1) = [(1/8 + 2/16)/2] \cdot [(1/8 + 1/16)/2] = 1/8 \cdot 3/32 = 3/256$
- $P(q|d_2) = [(1/8 + 2/16)/2] \cdot [(0/8 + 1/16)/2] = 1/8 \cdot 1/32 = 1/256$
- Ranking: $d_1 > d_2$

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

- LMs vs. vector space model: commonalities
  - Term frequency is directly in the model.
  - Probabilities are inherently "length-normalized".
  - Mixing document and collection frequencies has an effect similar to idf.
- LMs vs. vector space model: 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.

References

- *Introduction to Information Retrieval*, chapters 11 & 12.
- The slides were adapted from
  - Prof. Dragomir Radev's lectures at the University of Michigan:
    - http://clair.si.umich.edu/~radev/teaching.html
  - the book's companion website:
    - http://nlp.stanford.edu/IR-book/information-retrieval-book.html