



**Hasso  
Plattner  
Institut**

IT Systems Engineering | Universität Potsdam

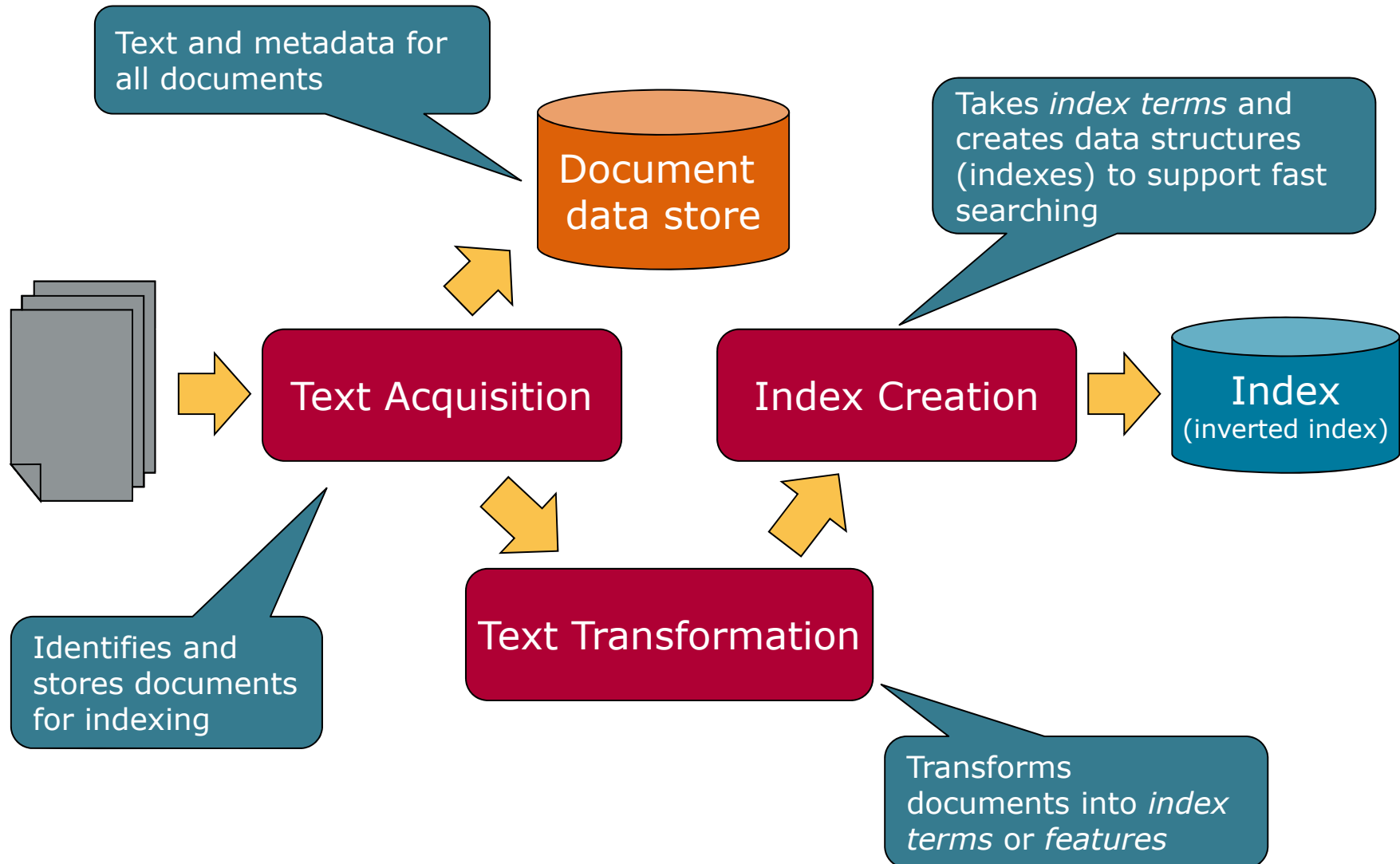
## Search Engines Chapter 7 – Retrieval Models

16.6.2009

Felix Naumann

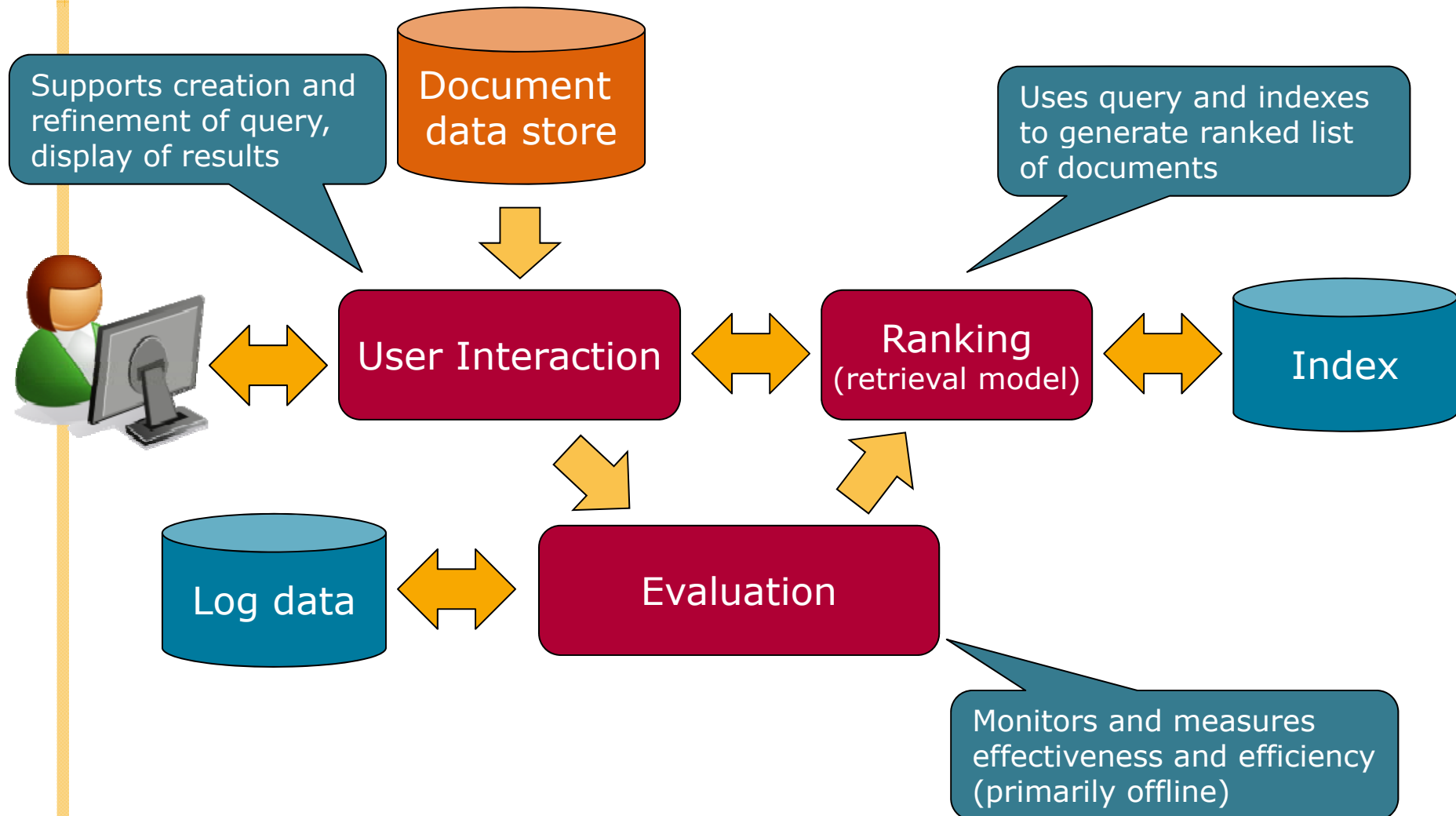
# The Indexing Process

2



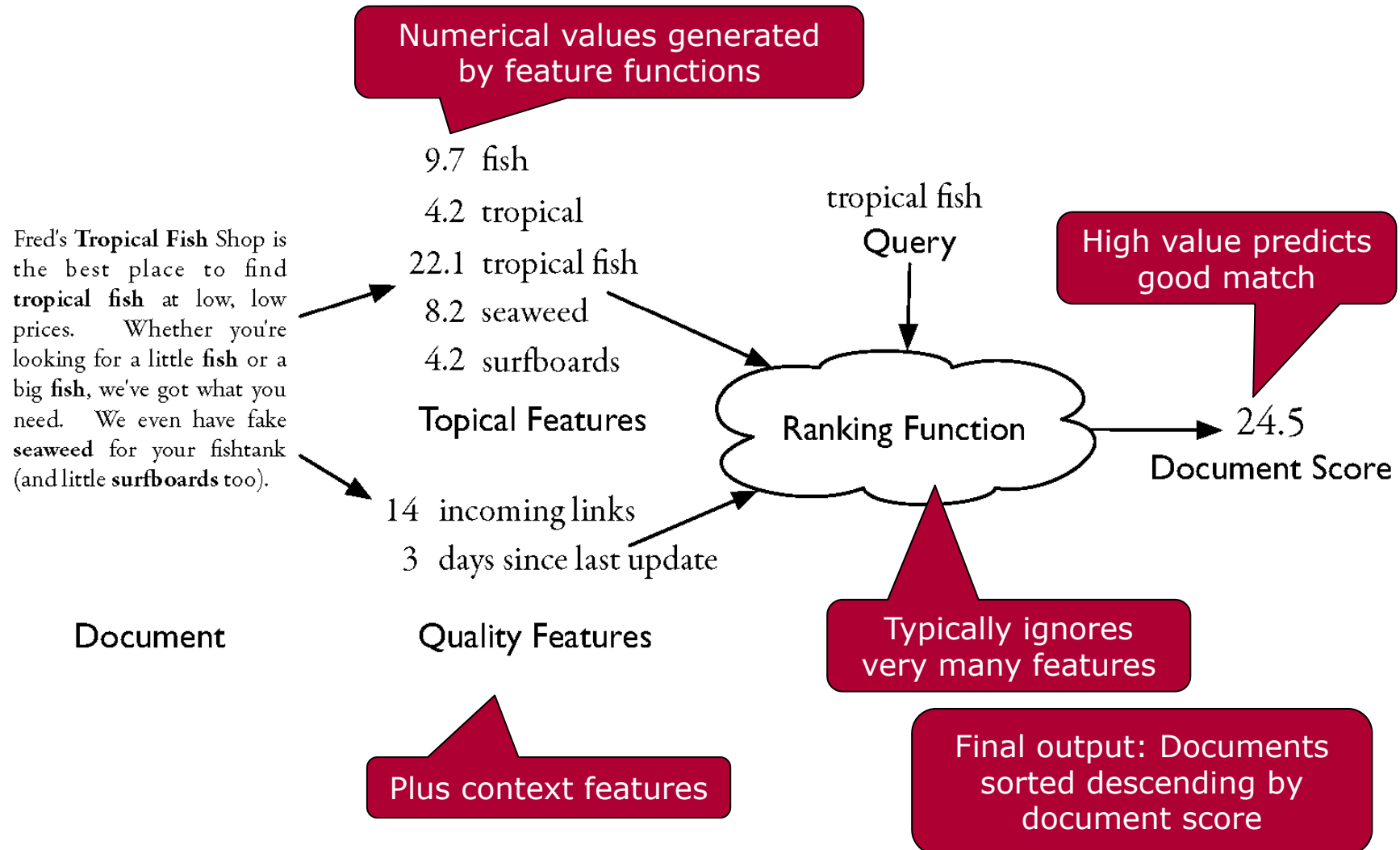
# The Query Process

3



# Abstract Model of Ranking

4



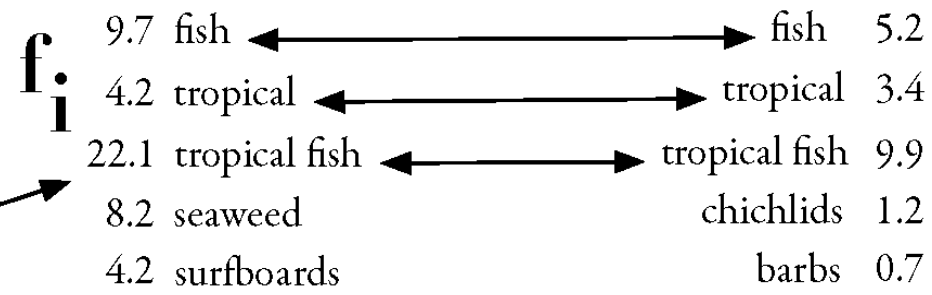
# More Concrete Model

5

$$R(Q, D) = \sum_i g_i(Q) f_i(D)$$

$f_i$  is a document feature function  
 $g_i$  is a query feature function

Fred's **Tropical Fish Shop** is the best place to find **tropical fish** at low, low prices. Whether you're looking for a little **fish** or a big **fish**, we've got what you need. We even have fake **seaweed** for your fishtank (and little **surfboards** too).



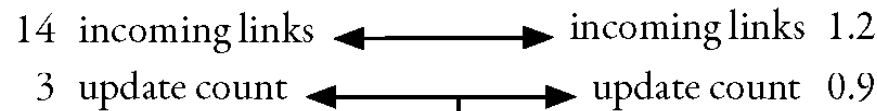
Topical Features

Topical Features

Only few; others are zero

**$g_i$**

tropical fish  
Query



Quality Features

Quality Features

Document

303.01  
Document Score

<http://www.howard.k12.md.us/res/aquariums/chichlids.html>



# Retrieval Models

6

- Provide a mathematical framework for defining the search process
  - Formalize human process of making decisions about relevance.
    - ◇ Framework should at least correlate well.
  - Includes explanation of assumptions
  - Basis of many ranking algorithms
  - Can be implicit
- Progress in retrieval models has corresponded with improvements in effectiveness.
  - Improvement of 100% in 90s (TREC)
- Mostly: Theories about relevance

# Relevance

7

- Complex concept that has been studied for some time
  - Many factors to consider
  - People often disagree when making relevance judgments.
    - ◇ Inter-annotator disagreement
- Retrieval models make various assumptions about relevance to simplify problem.
  - *Topical vs. user* relevance
    - ◇ Topical relevance: Document is of same topic
    - ◇ User relevance: All other factors
      - Some are used in some retrieval models
  - *Binary vs. multi-valued* relevance
    - ◇ Relevant vs. non-relevant
    - ◇ Relevant vs. unsure vs. non-relevant
    - ◇ Retrieval models usually are more detailed (probability)

# Overview

8

- ➔ ■ Older models
  - Boolean retrieval
  - Vector Space model
- Probabilistic models
- Language models
- Combining evidence
- Web search
- Learning to Rank





# Boolean Retrieval

9

- Two possible outcomes for query processing
  - TRUE or FALSE
  - “Exact-match” semantics
  - Simplest form of ranking
    - ◇ All matching documents are considered equally relevant.
- Query usually specified using Boolean operators
  - AND, OR, NOT
  - Proximity operators also used

# Boolean Retrieval

10

## ■ Advantages

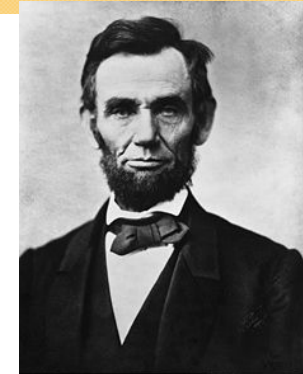
- Results are predictable and relatively easy to explain.
- Many different features can be incorporated
  - ◇ Date, document type, ...
- Efficient processing since many documents can be eliminated from search

## ■ Disadvantages

- Effectiveness depends entirely on user.
  - ◇ Presentation order not based on relevance
    - But arbitrarily on date, size, etc.
- Simple queries usually don't work well.
- Complex queries are difficult to write.
  - ◇ Search intermediaries (e.g. in legal offices)

# “Searching by Numbers”

11



- Sequence of queries driven by number of retrieved documents
  - Search of news articles for *President Lincoln*
    1. *lincoln*
      - ◇ Result: cars, places, people
    2. *president AND lincoln*
      - ◇ Result: “Ford Motor Company today announced that Darryl Hazel will succeed Brian Kelley as president of Lincoln Mercury.”
    3. *president AND lincoln AND NOT (automobile OR car)*
      - ◇ Not in result: “President Lincoln’s body departs Washington in a nine-car funeral train.”
    4. *president AND lincoln AND biography AND life AND birthplace AND gettysburg AND NOT (automobile OR car)*
      - ◇ Result: ∅
    5. *president AND lincoln AND (biography OR life OR birthplace OR gettysburg) AND NOT (automobile OR car)*
      - ◇ Top result might be: “President’s Day - Holiday activities - crafts, mazes, word searches, ... `The Life of Washington’ Read the entire book online! Abraham Lincoln Research Site ...”

# Vector Space Model

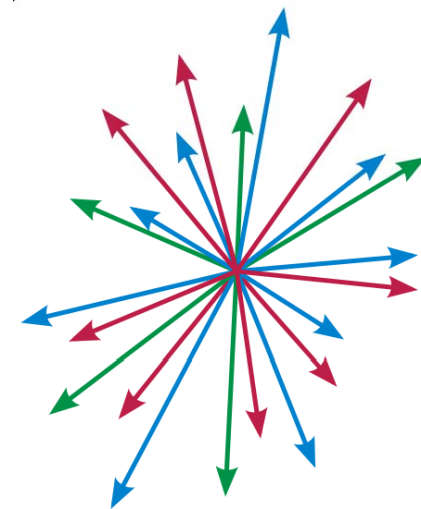
12

- Very popular model, even today
  - Simple, intuitive
  - Useful for weighting, ranking, and relevance feedback
- Documents and query represented by a vector of term weights
  - $t$  is number of index terms (i.e., very large)

$$D_i = (d_{i1}, d_{i2}, \dots, d_{it}) \quad Q = (q_1, q_2, \dots, q_t)$$

- Collection represented by a matrix of term weights

	$Term_1$	$Term_2$	$\dots$	$Term_t$
$Doc_1$	$d_{11}$	$d_{12}$	$\dots$	$d_{1t}$
$Doc_2$	$d_{21}$	$d_{22}$	$\dots$	$d_{2t}$
$\vdots$	$\vdots$			
$Doc_n$	$d_{n1}$	$d_{n2}$	$\dots$	$d_{nt}$



# Vector Space Model – Example

13

- $D_1$ : Tropical Freshwater Aquarium Fish.
- $D_2$ : Tropical Fish, Aquarium Care, Tank Setup.
- $D_3$ : Keeping Tropical Fish and Goldfish in Aquariums, and Fish Bowls.
- $D_4$ : The Tropical Tank Homepage - Tropical Fish and Aquariums.

Terms	Documents			
	$D_1$	$D_2$	$D_3$	$D_4$
aquarium	1	1	1	1
bowl	0	0	1	0
care	0	1	0	0
fish	1	1	2	1
freshwater	1	0	0	0
goldfish	0	0	1	0
homepage	0	0	0	1
keep	0	0	1	0
setup	0	1	0	0
tank	0	1	0	1
tropical	1	1	1	2

Rotated

Stopwords  
are removed

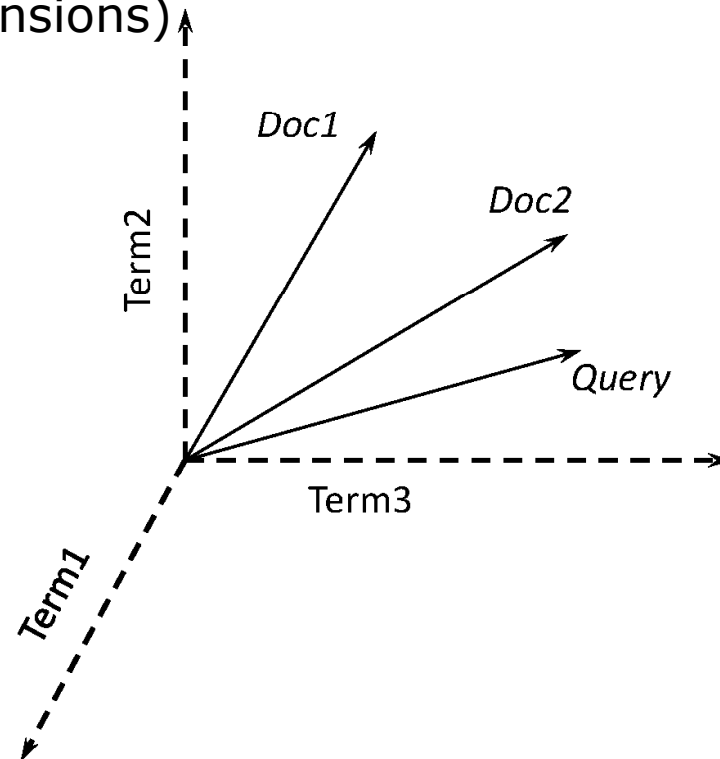
Weights are  
term counts

Query for „tropical fish“  
(0 0 0 1 0 0 0 0 0 1)

# Vector Space Model

14

- 3-d pictures useful, but can be misleading for high-dimensional space
  - Intuition no longer necessarily correct
  - Millions of terms (and dimensions)

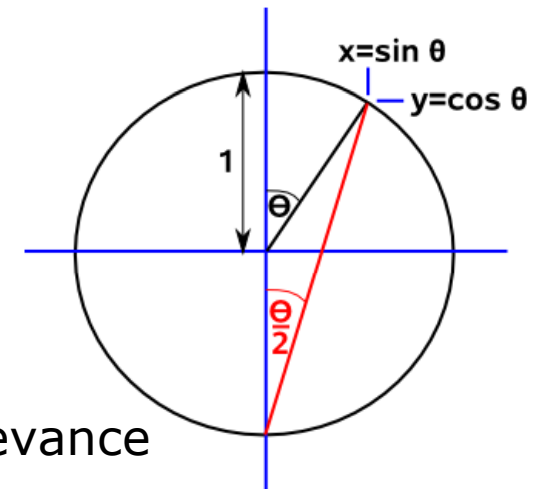


# Vector Space Model

15

- Each document ranked by distance between points representing query and document
  - *Similarity* measure more common than a distance or *dissimilarity* measure
  - Popular: Cosine correlation
    - ◇ Cosine of angle between document and query vectors
    - ◇ Normalized dot-product

$$\text{Cosine}(D_i, Q) = \frac{\sum_{j=1}^t d_{ij} \cdot q_j}{\sqrt{\sum_{j=1}^t d_{ij}^2 \cdot \sum_{j=1}^t q_j^2}}$$



- As retrieval model: No explicit definition of relevance
  - Implicit: Closer documents are more relevant.

<http://www.euclideanspace.com/maths/geometry/trig/derived/index.htm>

# Similarity Calculation – Example

16

- Consider two documents  $D_1, D_2$  and a query  $Q$ 
  - $D_1 = (0.5, 0.8, 0.3), D_2 = (0.9, 0.4, 0.2), D_3 = (0, 0.9, 0.1)$
  - $Q = (1.5, 1.0, 0)$
- Vector space model reflects term weights and number of matching terms (in contrast to Boolean retrieval)

$$\begin{aligned}
 \text{Cosine}(D_1, Q) &= \frac{(0.5 \times 1.5) + (0.8 \times 1.0)}{\sqrt{(0.5^2 + 0.8^2 + 0.3^2)(1.5^2 + 1.0^2)}} \\
 &= \frac{1.55}{\sqrt{(0.98 \times 3.25)}} = 0.87
 \end{aligned}$$

$$\begin{aligned}
 \text{Cosine}(D_2, Q) &= \frac{(0.9 \times 1.5) + (0.4 \times 1.0)}{\sqrt{(0.9^2 + 0.4^2 + 0.2^2)(1.5^2 + 1.0^2)}} \\
 &= \frac{1.75}{\sqrt{(1.01 \times 3.25)}} = 0.97 \qquad \text{Cosine}(D_3, Q) = 0.55
 \end{aligned}$$

- But: How to assign term weights?



# Term Weights

17

- *tf.idf* weight

- Term frequency weight measures importance in document *i*:

$$tf_{ik} = \frac{f_{ik}}{\sum_{j=1}^t f_{ij}}$$

- ◇ Long documents have many words with only one occurrence but also many with hundreds of occurrences
- ◇  $\log(f_{ik})$  to reduce this impact of frequent words

- Inverse document frequency measures importance in collection:  $idf_k = \log \frac{N}{n_k}$

- ◇ Reflects "amount of information" carried by term

- *tfidf* by multiplying *tf* and *idf* with some heuristic modifications

+1 to ensure non-zero weight

Normalization usually done by cosine similarity

$$d_{ik} = \frac{(\log(f_{ik})+1) \cdot \log(N/n_k)}{\sqrt{\sum_{k=1}^t [(\log(f_{ik})+1.0) \cdot \log(N/n_k)]^2}}$$

# Relevance Feedback – Rocchio algorithm

18

- Determine *Optimal query*
  - Maximizes the difference between the average vector representing the relevant documents and the average vector representing the non-relevant documents
- Usually only limited feedback (i.e., not for all documents): Only modify query
- Modifies query according to

$$q'_j = \alpha \cdot q_j + \beta \cdot \frac{1}{|Rel|} \sum_{D_i \in Rel} d_{ij} - \gamma \cdot \frac{1}{|Nonrel|} \sum_{D_i \in Nonrel} d_{ij}$$

- $q_j$  is initial term weight
- $Rel$  is set of relevant documents
- $Nonrel$  is set of non-relevant documents
  - ◇ Approximate as “all *unseen* documents”
- $\alpha$ ,  $\beta$ , and  $\gamma$  are parameters
  - ◇ Typical values 8, 16, 4
- New terms may be added (usually restricted to 50).
- Terms may accrue negative weight: Drop!

# Vector Space Model

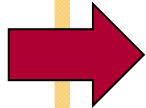
19

- Advantages
  - Simple computational framework for ranking
  - Any similarity measure or term weighting scheme could be used
    - ◇ Thus able to incorporate relevance feedback
- Disadvantages
  - Assumption of term independence
  - No *predictions* about techniques for effective ranking

# Overview

20

- Older models
- Probabilistic models
- Language models
- Combining evidence
- Web search
- Learning to Rank



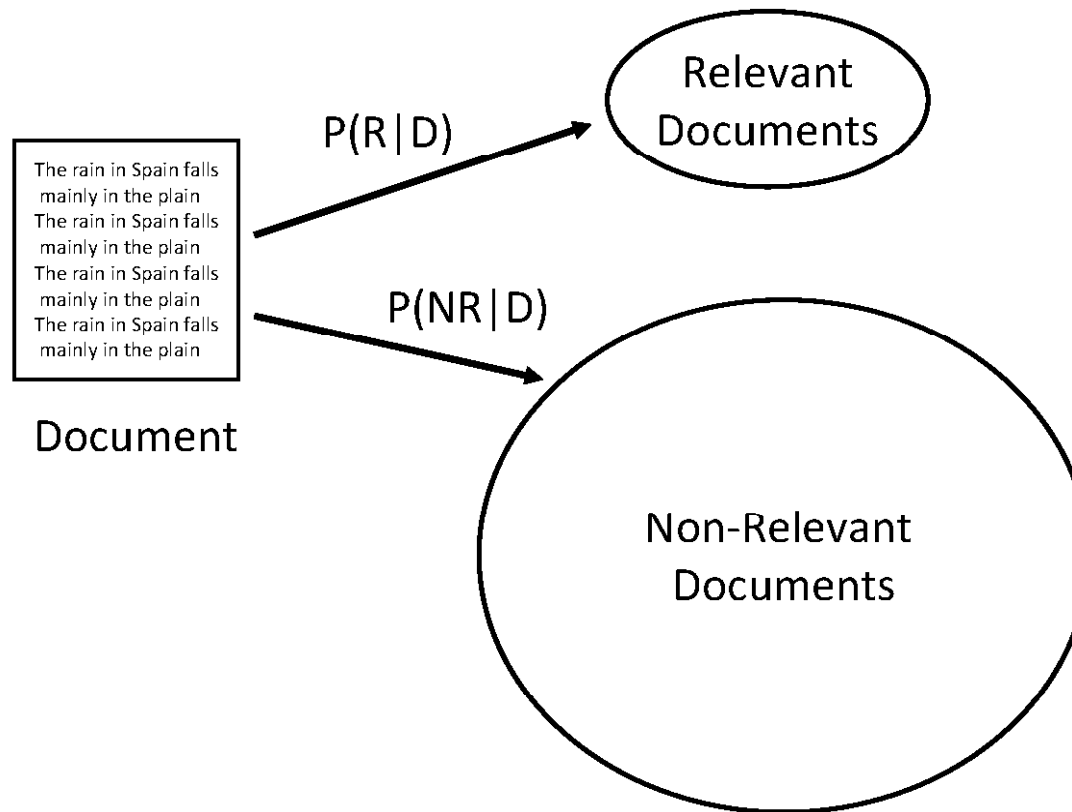
# Probability Ranking Principle

21

- Robertson (1977)
  - “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.”
- Probability theory is a strong foundation for representing and manipulating the inherent uncertainty
- Problem: How to estimate probability of relevance
  - Each model has different suggestion

# IR as Classification

22



Actually, we just need a ranking

# Bayes Classifier

23

- Bayes Decision Rule
  - A document  $D$  is relevant if  $P(R|D) > P(NR|D)$ .
- Estimating probabilities
  - Use Bayes Rule 
$$P(R|D) = \frac{P(D|R)P(R)}{P(D)}$$
  - Determining  $P(D|R)$  should be easier: Given information about the relevant set (e.g. relevant words/query), determine how likely it is to see the same properties in  $D$ .
- Example
  - Probability of “*president*” in relevant set is 0.02.
  - Probability of “*lincoln*” in relevant set is 0.03.
  - New document with “*president*” and “*lincoln*”. Probability of observing that combination is 0.0006.

# Bayes Classifier

24

- Bayes rule  $P(R|D) = \frac{P(D|R)P(R)}{P(D)}$ 
  - $P(R)$  is apriori probability of relevance (how likely is any document to be relevant)
  - $P(D)$  is normalizing constant.
- Before:  $D$  relevant if  $P(R|D) > P(NR|D)$ .
- Now: Classify a document as relevant if

$$\frac{P(D|R)}{P(D|NR)} > \frac{P(NR)}{P(R)}$$

- lhs is *likelihood ratio*
- Classification needs to make decision.
- Search engine only needs to rank.
  - Rank by likelihood ratio, ignore rhs





# Estimating $P(D|R)$

25

- *Binary independence model*

- Document represented as combinations of terms:
  - ◇ Vector of binary features indicating term occurrence (or non-occurrence)
- Represent  $R$  and  $NR$  as term-probabilities
  - ◇  $p_i$  is probability that term  $i$  occurs (i.e., has value 1) in relevant document,  $s_i$  is probability of occurrence in non-relevant document

- Assume independence (Naïve Bayes assumption)

$$P(D|R) = \prod_{i=1}^t P(d_i|R)$$

- Assumption is obviously incorrect, but successful

- Example:

- Document  $D$  contains words 1, 4, and 5:  $(1,0,0,1,1)$
- Let  $p_i$  denote probability that term  $i$  is in relevant set
- Relevance-probability of  $D$  is  $p_1 \times (1-p_2) \times (1-p_3) \times p_4 \times p_5$

# Binary Independence Model

26

- Let  $p_i$  denote probability that term  $i$  occurs in relevant set
- Let  $s_i$  denote probability that term  $i$  occurs in non-relevant set
- Reminder: Classify document as relevant if  $\frac{P(D|R)}{P(D|NR)} > \frac{P(NR)}{P(R)}$ 
  - Or rank according to lhs

$$\begin{aligned}
 \frac{P(D|R)}{P(D|NR)} &= \prod_{i:d_i=1} \frac{p_i}{s_i} \cdot \prod_{i:d_i=0} \frac{1-p_i}{1-s_i} \\
 &= \prod_{i:d_i=1} \frac{p_i}{s_i} \cdot \left( \prod_{i:d_i=1} \frac{1-s_i}{1-p_i} \cdot \prod_{i:d_i=1} \frac{1-p_i}{1-s_i} \right) \cdot \prod_{i:d_i=0} \frac{1-p_i}{1-s_i} \\
 &= \prod_{i:d_i=1} \frac{p_i(1-s_i)}{s_i(1-p_i)} \cdot \prod_i \frac{1-p_i}{1-s_i}
 \end{aligned}$$

# Binary Independence Model

27

$$\prod_{i:d_i=1} \frac{p_i(1-s_i)}{s_i(1-p_i)} \cdot \prod_i \frac{1-p_i}{1-s_i}$$

- Second term is over all documents, thus ignore
- To avoid accuracy problems, use log
- Scoring function is  $\sum_{i:d_i=1} \log \frac{p_i(1-s_i)}{s_i(1-p_i)}$
- Query provides information about relevant documents.
  - Summation only over terms that appear in query and document
- Simplification
  - If no further information about relevant set, assume  $p_i$  constant (e.g., 0.5)
  - Approximate  $s_i$  by entire collection (because number of relevant documents is very small).
  - Get *idf*-like weight
    - ◇ No *tf*-component, because binary features

$$\log \frac{0.5(1-\frac{n_i}{N})}{\frac{n_i}{N}(1-0.5)} = \log \frac{N-n_i}{n_i}$$

# Contingency Table

28

- If we do have information about term occurrences in relevant and non relevant information (through relevance feedback or pseudo-relevance feedback): Store in contingency table
  - $r_i$  is number of relevant documents containing term  $i$ .
  - $R$  is number of relevant documents for query.
  - $n_i$  is number of documents containing term  $i$ .
  - $N$  is total number of documents.

	Relevant	Non-relevant	Total
Term $i$ is present: $d_i = 1$	$r_i$	$n_i - r_i$	$n_i$
Term $i$ is not present: $d_i = 0$	$R - r_i$	$N - n_i - R + r_i$	$N - n_i$
Total	$R$	$N - R$	$N$

## Contingency Table

	Relevant	Non-relevant	Total
$d_i = 1$	$r_i$	$n_i - r_i$	$n_i$
$d_i = 0$	$R - r_i$	$N - n_i - R + r_i$	$N - n_i$
Total	$R$	$N - R$	$N$

29

- Idea: Use table to estimate  $p_i$  and  $s_i$  for scoring function

$$\sum_{i:d_i=1} \log \frac{p_i(1-s_i)}{s_i(1-p_i)}$$

- Obvious choices

- $p_i = r_i/R$
- $s_i = (n_i - r_i)/(N - R)$
- Problem if  $r_i = 0$
- Solution: Add 0.5 to counts and 1 to totals

$$p_i = (r_i + 0.5)/(R + 1)$$

$$s_i = (n_i - r_i + 0.5)/(N - R + 1)$$

- Gives scoring function:

$$\sum_{i:d_i=q_i=1} \log \frac{(r_i+0.5)/(R-r_i+0.5)}{(n_i-r_i+0.5)/(N-n_i-R+r_i+0.5)}$$

$$\sum_{i:d_i=q_i=1} \log \frac{(r_i+0.5)/(R-r_i+0.5)}{(n_i-r_i+0.5)/(N-n_i-R+r_i+0.5)}$$

- Uses only matching query terms
  - But: Relevance feedback can be used to expand query
- Not very good in practice
  - Missing *tf* component lowers effectiveness by 50%
  - I.e., 50% less relevant documents in top 10 compared to *tfidf* rankings
- But: Basis for BM25
  - Best Match variant 25

- Popular and effective ranking algorithm based on binary independence model
  - Adds document and query term weights
- Scoring function

$$\sum_{i \in Q} \log \frac{(r_i + 0.5) / (R - r_i + 0.5)}{(n_i - r_i + 0.5) / (N - n_i - R + r_i + 0.5)} \cdot \frac{(k_1 + 1) f_i}{K + f_i} \cdot \frac{(k_2 + 1) q f_i}{k_2 + q f_i}$$

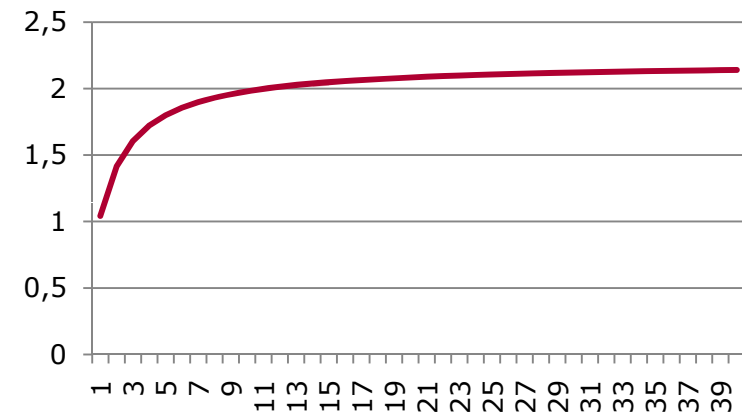
- Summation over all query terms
- $f_i$  is frequency of term  $i$  in document
- $qf_i$  is frequency of term  $i$  in query
- $k_1$ ,  $k_2$  and  $K$  are parameters whose values are set empirically.
- Reminders
  - ◇  $r_i$  is number of relevant documents containing term  $i$ .
    - Set to 0, if no relevance information
  - ◇  $R$  is number of relevant documents for query.
    - Set to 0, if no relevance information
  - ◇  $n_i$  is number of documents containing term  $i$ .
  - ◇  $N$  is total number of documents.

# BM25 – Interpretation

32

$$\sum_{i \in Q} \log \frac{(r_i + 0.5) / (R - r_i + 0.5)}{(n_i - r_i + 0.5) / (N - n_i - R + r_i + 0.5)} \cdot \frac{(k_1 + 1) f_i}{K + f_i} \cdot \frac{(k_2 + 1) q f_i}{k_2 + q f_i}$$

- $k_1$  determines how *tf* component of term weight changes as  $f_i$  increases
  - $k_1 = 0$ : term frequency ignored, only term presence
  - Typical:  $k_1 = 1.2$ , thus first few occurrences have most impact
- $k_2$  same for query term frequency
  - Typical:  $0 \leq k_2 \leq 1000$
  - Not sensitive, because low frequencies
- $K$  normalizes *tf* component by document length ( $dl$ ).
 
$$K = k_1 \left( (1 - b) + b \cdot \frac{dl}{avdl} \right)$$
  - $b$  regulates length normalization
    - ◇  $b = 0$ : No normalization
    - ◇  $b = 1$ : Full normalization
    - ◇ Typical:  $b = 0.75$



$$\frac{(k_1 + 1) f_i}{K + f_i} \quad k_1 = 1,2 \quad K = 1,1$$



# BM25 Example

33

- Query with two terms, “*president lincoln*”, ( $qf = 1$ )
- No relevance information:  $r = R = 0$
- $N = 500,000$  documents
- “*president*” occurs in 40,000 documents ( $n_1 = 40,000$ )
- “*lincoln*” occurs in 300 documents ( $n_2 = 300$ )
- “*president*” occurs 15 times in doc ( $f_1 = 15$ )
- “*lincoln*” occurs 25 times ( $f_2 = 25$ )
- Document length is 90% of the average length ( $dl/avdl = 0.9$ )
- $k_1 = 1.2$ ,  $b = 0.75$ , and  $k_2 = 100$
- $K = 1.2 \cdot (0.25 + 0.75 \cdot 0.9) = 1.11$

$$\sum_{i \in Q} \log \frac{(r_i + 0.5) / (R - r_i + 0.5)}{(n_i - r_i + 0.5) / (N - n_i - R + r_i + 0.5)} \cdot \frac{(k_1 + 1) f_i}{K + f_i} \cdot \frac{(k_2 + 1) q f_i}{k_2 + q f_i}$$

# BM25 Example

34

$$\sum_{i \in Q} \log \frac{(r_i + 0.5) / (R - r_i + 0.5)}{(n_i - r_i + 0.5) / (N - n_i - R + r_i + 0.5)} \cdot \frac{(k_1 + 1) f_i}{K + f_i} \cdot \frac{(k_2 + 1) q f_i}{k_2 + q f_i}$$

$$BM25(Q, D) =$$

*president*

$$\log \frac{(0 + 0.5) / (0 - 0 + 0.5)}{(40000 - 0 + 0.5) / (500000 - 40000 - 0 + 0 + 0.5)} \times \frac{(1.2 + 1) 15}{1.11 + 15} \times \frac{(100 + 1) 1}{100 + 1}$$

*lincoln*

$$+ \log \frac{(0 + 0.5) / (0 - 0 + 0.5)}{(300 - 0 + 0.5) / (500000 - 300 - 0 + 0 + 0.5)} \times \frac{(1.2 + 1) 25}{1.11 + 25} \times \frac{(100 + 1) 1}{100 + 1}$$

Without relevance, first factor is similar to idf: 2.44 for *president*, 7.42 for *lincoln*.

$$\begin{aligned} &= \log 460000.5 / 40000.5 \cdot 33 / 16.11 \cdot 101 / 101 \\ &+ \log 499700.5 / 300.5 \cdot 55 / 26.11 \cdot 101 / 101 \\ &= 2.44 \cdot 2.05 \cdot 1 + 7.42 \cdot 2.11 \cdot 1 \\ &= 5.00 + 15.66 = 20.66 \end{aligned}$$

F

# BM25 Example

35

- Effect of term frequencies

Frequency of “president”	Frequency of “lincoln”	BM25 score
15	25	20.66
15	1	12.74
15	0	5.00
1	25	18.2
0	25	15.66

- Even one occurrence of *lincoln* makes for a large difference in score.
  - Occurrence of president less important
- Document with very many occurrences of one word can be better than one with both words.
  - 15.66 > 12.74

# BM25 – Discussion

Contant per term

Contant per query

Hasso Plattner Institut

36

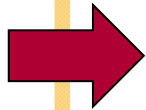
$$\sum_{i \in Q} \log \frac{(r_i + 0.5) / (R - r_i + 0.5)}{(n_i - r_i + 0.5) / (N - n_i - R + r_i + 0.5)} \cdot \frac{(k_1 + 1) f_i}{K + f_i} \cdot \frac{(k_2 + 1) q f_i}{k_2 + q f_i}$$

- Seems complicated, but
  - Calculation of term weights at index time
  - With no relevance info, just add weights for matching query terms
    - ◇ Plus some additional calculation for multiple query terms ( $qf > 1$ )
- Well tuneable to different applications

# Overview

37

- Older models
- Probabilistic models
- Language models
- Combining evidence
- Web search
- Learning to Rank



# Language Model

38

- Language model applications
  - Speech recognition, machine translation, handwriting recognition
  - And information retrieval
- Predicts which word is next in a sequence of words.
- *Unigram language model*
  - Probability distribution over the words in a language
  - Generation of text consists of pulling words out of a “bucket” according to the probability distribution and replacing them.
  - Next word not dependent on previous word(s).
  - Example for language with 5 words: (.2, .1, .35, .25, .1)
- *N-gram language model*
  - Predicts next word based on previous  $n-1$  words.
  - Some applications use bigram and trigram language models where probabilities depend on previous words.
- Bi- and tri-grams expensive – unigram models suffice for search applications.

# Language Model

39

- A *topic* in a document can be represented as a language model
  - i.e., as a distribution over words.
  - Words that tend to occur often when discussing a topic will have high probabilities in the corresponding language model
  - In general: Distribution over all words, but most (unimportant words) will have default probability.
- *Multinomial* distribution over words
  - Text is modeled as a finite sequence of words, where there are  $t$  possible words at each point in the sequence.
  - Commonly used, but not only possibility
  - Does not model *burstiness*
    - ◇ Occurrence of a word makes repeated occurrence more likely
  - Not here...
- The topic of a query can also be represented as language model.

# LMs for Retrieval

40

- Three possibilities to use language models for retrieval:
  1. Probability of generating the query text from a document language model
  2. Probability of generating the document text from a query language model
  3. Comparing the language models representing the query and document topics
- Models of topical relevance
  - Query-Likelihood Model
  - Relevance model / document-likelihood model



# Query-Likelihood Model

41

- Rank documents by the probability that the query could be generated by the document model
  - Probability that we could pull the query words from the bucket of document words
  - i.e., same topic
- Given query, start with  $P(D|Q)$
- Using Bayes' Rule, ignoring normalizing constant  $P(Q)$

$$p(D|Q) \stackrel{rank}{=} P(Q|D)P(D)$$

- Assuming prior is uniform, unigram model

$$P(Q|D) = \prod_{i=1}^n P(q_i|D)$$

- Possible non-uniform prior: Use date or document length

# Estimating Probabilities

42

- Obvious estimate for unigram probabilities is

$$P(q_i|D) = \frac{f_{q_i,D}}{|D|}$$

- *Maximum likelihood estimate*
  - makes the observed value of  $f_{q_i,D}$  most likely
- Problems:
  - If 1 query word out of 6 is missing from document, score will be zero
  - Missing 1 out of 6 query words same as missing 5 out of 6
  - Words associated with topic should have some probability, even if they do not appear in document.
    - ◇ Assign at least some small probability
- Thus: Smoothing

# Smoothing

43

- Document texts are a *sample* from the language model
  - Missing words should not have zero probability of occurring
- *Smoothing* is a technique for estimating probabilities for missing (or unseen) words.
  - Lower (or *discount*) the probability estimates for words that are seen in the document text.
  - Assign that “left-over” probability to the estimates for the words that are not seen in the text.
    - ◇ Usually based on frequency of words in entire collection of documents

# Estimating Probabilities

44

- Estimate for unseen words is  $a_D P(q_i|C)$ 
  - $P(q_i|C)$  is the probability for query word  $i$  in the *collection* language model for collection  $C$  (background probability)
  - $a_D$  is a parameter between 0 and 1
- Estimate for words that occur is  $(1 - a_D) P(q_i|D) + a_D P(q_i|C)$ 
  - To ensure summation to 1
- Different forms of estimation come from different  $a_D$
- Example: Only three words in collection  $w_1, w_2, w_3$ 
  - $P(w_1|C) = 0.3$      $P(w_2|C) = 0.5$      $P(w_3|C) = 0.2$
  - $P(w_1|D) = 0.5$      $P(w_2|D) = 0.5$      $P(w_3|D) = 0$
  - Smoothing
    - ◇  $P(w_1|D) = (1 - a_D) P(w_1|D) + a_D P(w_1|C)$   
 $= (1 - a_D) 0.5 + a_D 0.3$
    - ◇  $P(w_2|D) = (1 - a_D) 0.5 + a_D 0.5$
    - ◇  $P(w_3|D) = (1 - a_D) 0.0 + a_D 0.2$     ( $= a_D 0.2 > 0$  !)
    - ◇ Test:  $P(w_1|D) + P(w_2|D) + P(w_3|D) = 1$
- Variations based on different choices for  $a_D$

# Jelinek-Mercer Smoothing

45

- Simple choice:  $a_D$  is a constant,  $a_D = \lambda$
- Gives estimate of 
$$p(q_i|D) = (1 - \lambda) \frac{f_{q_i,D}}{|D|} + \lambda \frac{c_{q_i}}{|C|}$$
- Ranking score 
$$P(Q|D) = \prod_{i=1}^n \left( (1 - \lambda) \frac{f_{q_i,D}}{|D|} + \lambda \frac{c_{q_i}}{|C|} \right)$$
- Use logs for convenience
  - Due to accuracy problems when multiplying many small numbers
$$\log P(Q|D) = \sum_{i=1}^n \log \left( (1 - \lambda) \frac{f_{q_i,D}}{|D|} + \lambda \frac{c_{q_i}}{|C|} \right)$$
- Small  $\lambda$  result in less smoothing, closer to Boolean AND
  - $\lambda = 0.1$  successful for short queries
- For high  $\lambda$  relative weighting less important, closer to Boolean OR
  - Coordination level match: Ranks by number of matching query terms
  - $\lambda = 0.7$  successful for very long queries

$$\log_a \prod_{i=1}^n x_i = \sum_{i=1}^n \log_a x_i.$$

# Where is *tf.idf*-like weight?

46

$$\log P(Q|D) = \sum_{i=1}^n \log\left((1 - \lambda) \frac{f_{q_i,D}}{|D|} + \lambda \frac{c_{q_i}}{|C|}\right)$$

$$= \sum_{i:f_{q_i,D} > 0} \log\left((1 - \lambda) \frac{f_{q_i,D}}{|D|} + \lambda \frac{c_{q_i}}{|C|}\right) + \sum_{i:f_{q_i,D} = 0} \log\left(\lambda \frac{c_{q_i}}{|C|}\right)$$

$$= \sum_{i:f_{q_i,D} > 0} \log \frac{\left((1 - \lambda) \frac{f_{q_i,D}}{|D|} + \lambda \frac{c_{q_i}}{|C|}\right)}{\lambda \frac{c_{q_i}}{|C|}} + \sum_{i=1}^n \log\left(\lambda \frac{c_{q_i}}{|C|}\right)$$

$$\stackrel{rank}{=} \sum_{i:f_{q_i,D} > 0} \log \left( \frac{\left((1 - \lambda) \frac{f_{q_i,D}}{|D|} + 1\right)}{\lambda \frac{c_{q_i}}{|C|}} \right)$$

Split into words that occur and those that do not

Add and subtract  $\sum_{i:f_{q_i,D} > 0} \log\left(\lambda \frac{c_{q_i}}{|C|}\right)$

Same for all documents: Ignore

proportional to the term frequency

inversely proportional to the collection frequency

# Dirichlet Smoothing

47

- More effective choice: let  $a_D$  depend on document length:

$$\alpha_D = \frac{\mu}{|D| + \mu}$$

- Substituted in  $(1 - a_D) P(q_i|D) + a_D P(q_i|C)$  gives probability estimation

$$p(q_i|D) = \frac{f_{q_i, D} + \mu \frac{c_{q_i}}{|C|}}{|D| + \mu}$$

- and document score

$$\log P(Q|D) = \sum_{i=1}^n \log \frac{f_{q_i, D} + \mu \frac{c_{q_i}}{|C|}}{|D| + \mu}$$

- Small values for  $\mu$  give more importance to relative term weights.
- Large values favor number of matching terms.
- Typical:  $1,000 \leq \mu \leq 2,000$

# Query Likelihood Example

48

- For the term "president"
  - $f_{qi,D} = 15, c_{qi} = 160,000$
- For the term "lincoln"
  - $f_{qi,D} = 25, c_{qi} = 2,400$
- Number of word occurrences in the document  $|d|$  is assumed to be 1,800.
- Number of word occurrences in the collection is  $10^9$ .
  - 500,000 documents times an average of 2,000 words
- $\mu = 2,000$

$$\begin{aligned}
 QL(Q, D) &= \log \frac{15 + 2000 \times (1.6 \times 10^5 / 10^9)}{1800 + 2000} \\
 &\quad + \log \frac{25 + 2000 \times (2400 / 10^9)}{1800 + 2000} \\
 &= \log(15.32 / 3800) + \log(25.005 / 3800) \\
 &= -5.51 + -5.02 = -10.53
 \end{aligned}$$

• Negative number because summing logs of small numbers  
 • Only ranking is relevant




# Query Likelihood Example

49


QL:

Frequency of "president"	Frequency of "lincoln"	QL score
15	25	-10.53
15	1	-13.75
15	0	-19.05
1	25	-12.99
0	25	-14.40



BM25:

Frequency of "president"	Frequency of "lincoln"	BM25 score
15	25	20.66
15	1	12.74
15	0	5.00
1	25	18.2
0	25	15.66



# Query Likelihood Discussion

50

- Simple probabilistic retrieval model
- Uses probability estimations as term weights
- QL with Dirichlet smoothing similar to BM25
- QL with advanced smoothing consistently better than BM25
  - Advanced smoothing: Use only similar documents instead of entire collection. Later...
  
- Disadvantages
  - Difficult to incorporate information about relevant documents into ranking
  - Difficult to represent the fact that a query is just one of many possible queries to describe a particular need

# Relevance Models

51

- Represent topic of query as language model
  - Call this the *relevance model* – language model representing information need
  - Query: Very small sample generated from this model
  - Relevant documents: Larger samples from same model
- $P(D|R)$  - probability of generating the text in a document given a relevance model
  - *Document likelihood* model
  - Less effective than query likelihood due to
    - ◇ Large and extremely variable number of words
    - ◇ Difficulties comparing across documents of different lengths
      - $|D_a| = 5$ ;  $|D_b| = 500$
      - $P(D_a|R)$  and  $P(D_b|R)$  vs.  $P(Q|D_a)$  and  $P(Q|D_b)$
    - ◇ Difficult to obtain relevance model (examples for relevant documents)

# Pseudo-Relevance Feedback

52

- Idea:

1. Estimate relevance model from query and top-ranked documents.
2. Rank documents by similarity of document model to relevance model
  - ◇ *Kullback-Leibler divergence* (KL-divergence) is a well-known measure of the difference between two probability distributions

# KL-Divergence

53

- Given the *true* probability distribution  $P$  and another distribution  $Q$  that is an *approximation* to  $P$ ,

$$KL(P||Q) = \sum_x P(x) \log \frac{P(x)}{Q(x)}$$

- Divergence: Large values mean large difference, mean low similarity.
- $KL(P||Q) \geq 0$
- Not symmetric:  $KL(P||Q) \neq KL(Q||P)$ 
  - ◇ Choice of “true” distribution is important.
- Use negative KL-divergence for ranking, and assume relevance model  $R$  is the true distribution:

$$\sum_{w \in V} P(w|R) \log P(w|D) - \sum_{w \in V} P(w|R) \log P(w|R)$$

- Summation over all words in vocabulary

$$\log_a \frac{x}{y} = \log_a x - \log_a y$$

# KL-Divergence

54

$$\sum_{w \in V} P(w|R) \log P(w|D) - \sum_{w \in V} P(w|R) \log P(w|R)$$

- Second term same for each document: Ignore for ranking
- Given a simple maximum likelihood estimate for  $P(w|R)$ , based on the frequency in the query text, ranking score is

$$\sum_{w \in V} \frac{f_{w,Q}}{|Q|} \log P(w|D)$$

- This is rank-equivalent to query likelihood score.
  - Non-query words are iterated but contribute zero.
  - Query words with frequency  $k$  contribute  $k$  times  $\log P(w|D)$ .
- Query likelihood model is a special case of retrieval based on relevance model
  - More general model allows more sophisticated estimation based on other query words. Now...

# Estimating the Relevance Model

55

- Probability of pulling a word  $w$  out of the “bucket” representing the relevance model depends on the  $n$  query words we have just pulled out

$$P(w|R) \approx P(w|q_1 \dots q_n)$$

- By definition

$$P(w|R) \approx \frac{P(w, q_1 \dots q_n)}{P(q_1 \dots q_n)}$$

- $P(q_1, \dots, q_n)$  is normalizing constant

$$P(q_1 \dots q_n) = \sum_{w \in V} P(w, q_1 \dots q_n)$$

- Now: Estimate  $P(w, q_1, \dots, q_n)$

## Estimating the Relevance Model

56

- Given document set  $\mathcal{C}$  represented by language models, joint probability is

$$P(w, q_1 \dots q_n) = \sum_{D \in \mathcal{C}} p(D) P(w, q_1 \dots q_n | D)$$

- Assume

$$P(w, q_1 \dots q_n | D) = P(w | D) \prod_{i=1}^n P(q_i | D)$$

- Gives

$$P(w, q_1 \dots q_n) = \sum_{D \in \mathcal{C}} P(D) P(w | D) \prod_{i=1}^n P(q_i | D)$$



# Estimating the Relevance Model

57

$$P(w, q_1 \dots q_n) = \sum_{D \in \mathcal{C}} P(D) P(w|D) \prod_{i=1}^n P(q_i|D)$$

- $P(D)$  usually assumed to be uniform: Ignore
- $\prod_{i=1}^n P(q_i|D)$  is query likelihood score for  $D$ .
  - Thus,  $P(w, q_1 \dots q_n)$  is simply a weighted average of the language model probabilities for  $w$  in a set of documents, where the weights are the query likelihood scores for those documents.
- We are adding words to query by smoothing relevance model using documents that are similar to query.
- This is precisely a formal model for pseudo-relevance feedback
  - Used as query expansion technique: Words with zero weight in relevance model will now have non-zero weights

# Pseudo-Feedback Algorithm

58

1. Rank documents using the query likelihood score for query  $Q$ .
  - Use Dirichlet-smoothing for  $P(w|D)$
2. Select number of the top-ranked documents to be the set  $C$ .
  - Using entire collection including low-ranked documents would not be helpful. Also: Faster calculation

3. Calculate the relevance model probabilities  $P(w|R)$  using
 
$$P(w|R) \approx \frac{P(w, q_1 \dots q_n)}{P(q_1 \dots q_n)}$$

$$P(w, q_1 \dots q_n) = \sum_{D \in C} P(D) P(w|D) \prod_{i=1}^n P(q_i|D)$$

- $P(q_1 \dots q_n)$  is used as a normalizing constant and is calculated as before as

$$P(q_1 \dots q_n) = \sum_{w \in V} P(w, q_1 \dots q_n)$$

4. Rank documents again using the KL-divergence score:
 
$$\sum_w P(w|R) \log P(w|D)$$
  - Use Dirichlet-smoothing for  $P(w|D)$
  - Iterate only over highest-probability words for efficiency

## Example from Top 10 Docs

Strong focus on source type (news)  
This will reflect results of pseudo-relevance feedback

59

<i>president lincoln</i>	<i>abraham lincoln</i>	<i>fishing</i>	<i>tropical fish</i>
lincoln	lincoln	fish	fish
president	america	farm	tropic
room	president	salmon	japan
bedroom	faith	new	aquarium
house	guest	wild	water
white	abraham	water	species
america	new	caught	aquatic
guest	room	catch	fair
serve	christian	tag	china
bed	history	time	coral
washington	public	eat	source
old	bedroom	raise	tank
office	war	city	reef
war	politics	people	animal
long	old	fishermen	tarpon
abraham	national	boat	fishery

16 highest-probability words from relevance model

## Example from Top 50 Docs

More general, because larger variety of topics in documents

60

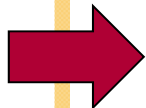
<i>president lincoln</i>	<i>abraham lincoln</i>	<i>fishing</i>	<i>tropical fish</i>
lincoln	lincoln	fish	fish
president	president	water	tropic
america	america	catch	water
new	abraham	reef	storm
national	war	fishermen	species
great	man	river	boat
white	civil	new	sea
war	new	year	river
washington	history	time	country
clinton	two	bass	tuna
house	room	boat	world
history	booth	world	million
time	time	farm	state
center	politics	angle	time
kennedy	public	fly	japan
room	guest	trout	mile

16 highest-probability words from relevance model

# Overview

61

- Older models
- Probabilistic models
- Language models
- Combining evidence
- Web search
- Learning to Rank



# Combining Evidence

62

- Effective retrieval requires the combination of many pieces of evidence about a document's potential relevance.
  - Until now: focus on simple word-based evidence
  - Many other types of evidence
    - ◇ Words: Structure, proximity of word, relationships among words
    - ◇ Metadata: PageRank, publication date, document type
    - ◇ Scores from different models
- Variant 1: Adapt BM25 or Query Likelihood with additional factors
  - Difficult to maintain, understand and tune
- Variant 2: *Inference network* model is one approach to combining evidence
  - Probabilistic model
  - Uses Bayesian network formalism
  - Mechanism to define and evaluate operators in a query language
    - ◇ Operators to specify evidence
    - ◇ Operators to combine evidence

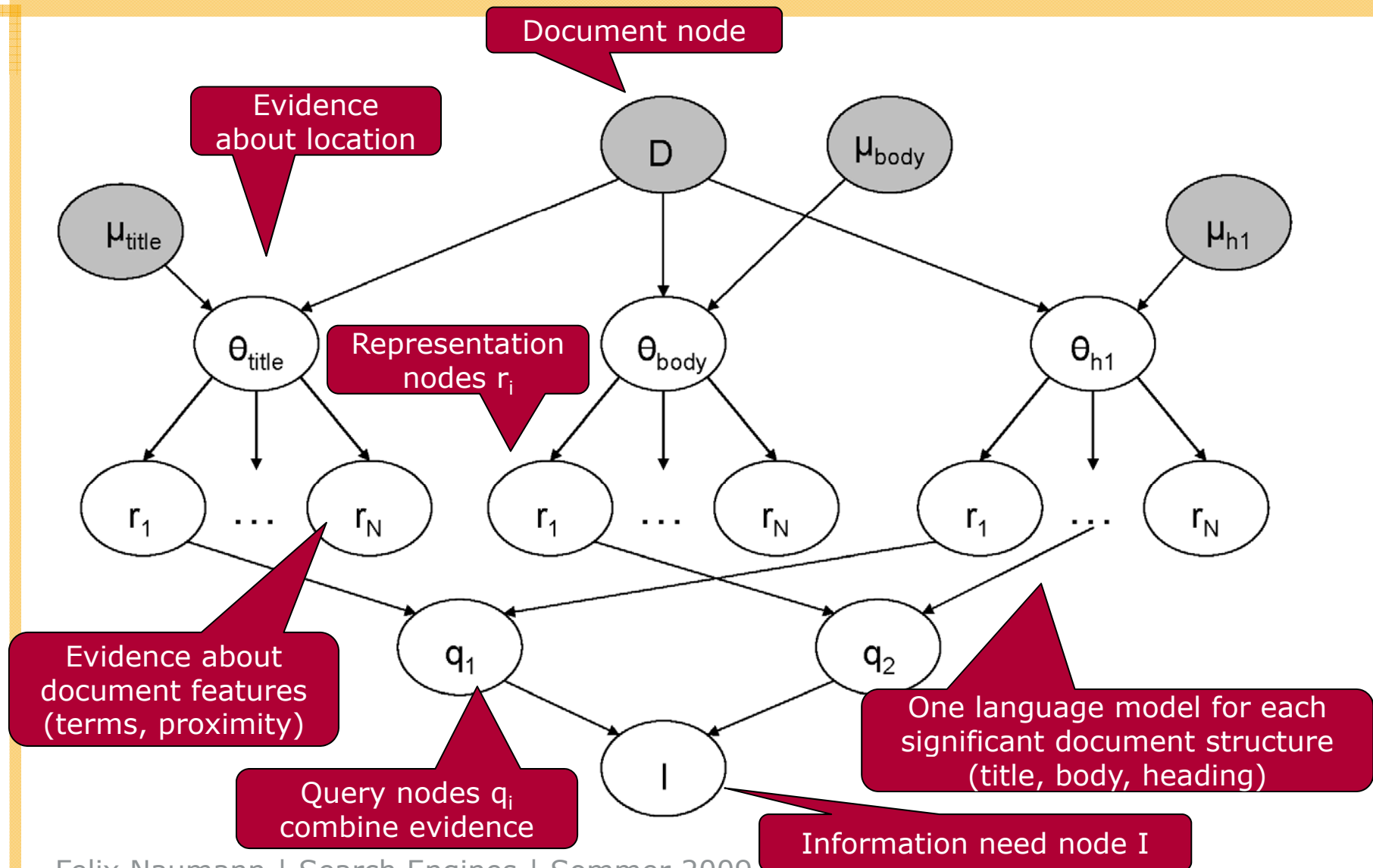
# Bayesian Networks

63

- Probabilistic model
- Specifies set of events and dependencies between them
- Modeled as DAG – directed acyclic graph
  - Nodes: Events
    - ◇ Here: Observing a particular document or piece of evidence or some combination of evidences
    - ◇ All binary
  - Arcs: probabilistic dependencies between events

# Inference Network

64

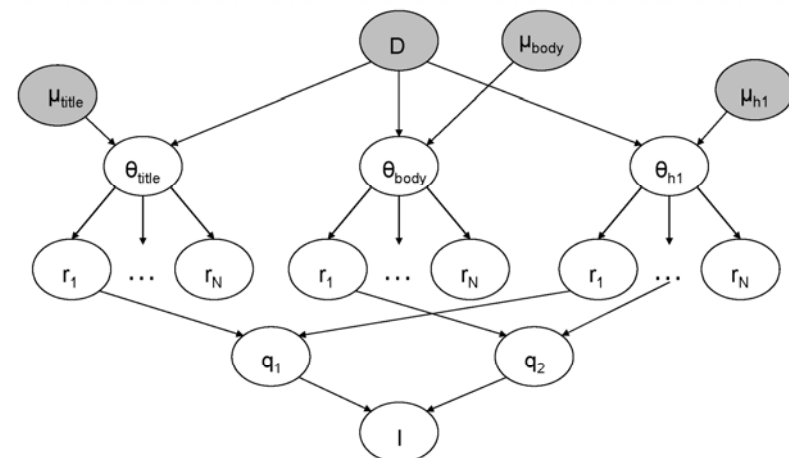




# Inference Network

65

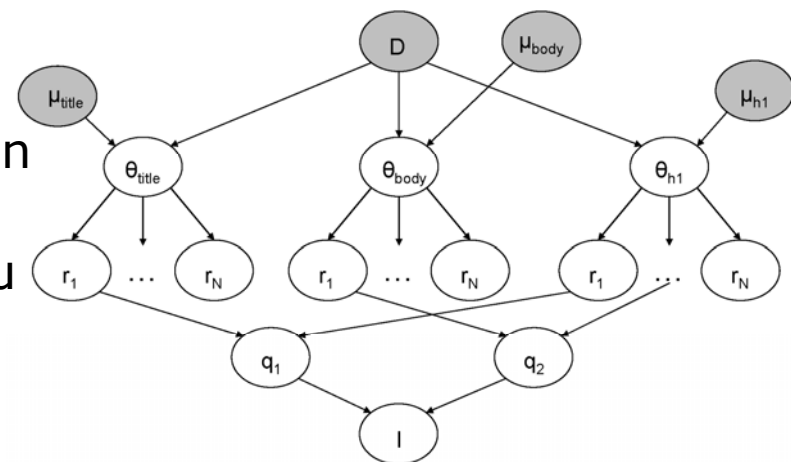
- *Document node* ( $D$ ) corresponds to the event that a document is observed.
- *Representation nodes* ( $r_i$ ) are document features (evidence)
  - Probabilities associated with those features are based on language models  $\theta$  estimated using the parameters  $\mu$
  - One language model for each significant document structure
  - $r_i$  nodes can represent proximity features, or other types of evidence, e.g., date



# Inference Network

66

- *Query nodes* ( $q_i$ ) are used to combine evidence from representation nodes and other query nodes
  - Represent the occurrence of more complex evidence and document features
  - A number of combination operators are available
    - ◇ AND, OR, ...
- *Information need node* ( $I$ ) is a special query node that combines all of the evidence from the other query nodes
  - In all, network computes  $P(I|D, \mu)$
  - = probability that an information need is met given the document and the parameters  $\mu$
  - Used to rank documents



# Inference Network

67

- Connections in inference network defined by query and by representation nodes
- Probabilities for representation nodes estimated using language model
  - Reflect probability that feature is characteristic of document
    - ◇ Not probability of occurrence
  - Node for „lincoln“ represents binary event that document is about that topic.
  - Language model used to calculate probability that that event is TRUE.
- Document is represented by binary vector

# Inference Network

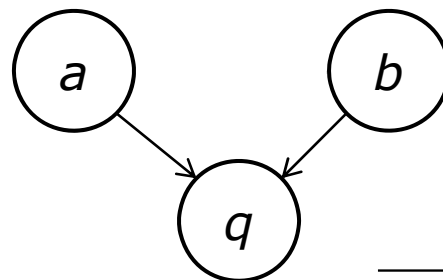
68

- To calculate probabilities: 
$$P(r_i|D, \mu) = \frac{f_{r_i,D} + \mu P(r_i|C)}{|D| + \mu}$$
  - Same as before – Dirichlet smoothing
  - $f_{i,D}$  is number of times feature  $r_i$  occurs in  $D$
  - $P(r_i|C)$  is collection probability for feature  $r_i$
  - $\mu$  is Dirichlet smoothing parameter
    - ◇ Specific to the document structure of interest
- Example:  $f_{i,D}$  is number of times „lincoln“ appears in title
  - Collection probability calculated based on all collection titles
  - $\mu$  is title-specific

# Example: AND Combination

69

- Query nodes are basis for operators of query language
  - Restricted to combinations that can be efficiently calculated
  - Calculate probability of each outcome (true or false) given all possible states of parent nodes
- Example for Boolean AND:



*a and b are parent nodes for q*

$P(q = \text{TRUE}   a, b)$	<i>a</i>	<i>b</i>
0	FALSE	FALSE
0	FALSE	TRUE
0	TRUE	FALSE
1	TRUE	TRUE

## Example: AND Combination

70

- Combination must consider all possible states of parents
- Some combinations can be computed efficiently
- Let  $p_{xy}$  denote probability that  $q$  is TRUE given state  $x$  and  $y$  of parents.
  - $p_a$  is probability that  $a$  is TRUE
- Calculate *belief value* (probability) from an AND combination:

$$\begin{aligned}
 \text{bel}_{and}(q) &= p_{00}P(a = \text{FALSE})P(b = \text{FALSE}) \\
 &\quad + p_{01}P(a = \text{FALSE})P(b = \text{TRUE}) \\
 &\quad + p_{10}P(a = \text{TRUE})P(b = \text{FALSE}) \\
 &\quad + p_{11}P(a = \text{TRUE})P(b = \text{TRUE}) \\
 &= 0 \cdot (1 - p_a)(1 - p_b) + 0 \cdot (1 - p_a)p_b + 0 \cdot p_a(1 - p_b) + 1 \cdot p_a p_b \\
 &= p_a p_b
 \end{aligned}$$

# Inference Network Operators

Unary operator

71

- Other operators can also be calculated efficiently.
- Let  $q$  have  $n$  parents,
  - each with probability  $p_i$  of being true.

$wt_i$  is weight of parent to indicate relative importance

$$bel_{not}(q) = 1 - p_1$$

$$bel_{or}(q) = 1 - \prod_i^n (1 - p_i)$$

$$bel_{and}(q) = \prod_i^n p_i$$

$$bel_{wand}(q) = \prod_i^n p_i^{wt_i}$$

$$bel_{max}(q) = \max\{p_1, p_2, \dots, p_n\}$$

$$bel_{sum}(q) = \frac{\sum_i^n p_i}{n}$$

$$bel_{wsum}(q) = \frac{\sum_i^n wt_i p_i}{\sum_i^n wt_i}$$

# Galago Query Language

72

- Given description of underlying model and combination operators, we can define a query language that can be used in a search engine to produce rankings based on complex combinations of evidence.
- Example here: Galago (galagosearch.org, Developed by authors of textbook)
- Query: „pet therapy“ compiled to Galago query

```
#weight(
0.1 #weight( 0.6 #prior(pagerank) 0.4 #prior(inlinks))
1.0 #weight(
    0.9 #combine(
        #weight(1.0 pet.(anchor) 1.0 pet.(title)
            3.0 pet.(body) 1.0 pet.(heading))
        #weight(1.0 therapy.(anchor) 1.0 therapy.(title)
            3.0 therapy.(body) 1.0 therapy.(heading)))
0.1 #weight(
    1.0 #od1(pet therapy).(anchor)
    1.0 #od1(pet therapy).(title)
    3.0 #od1(pet therapy).(body)
    1.0 #od1(pet therapy).(heading))
0.1 #weight(
    1.0 #uw8(pet therapy).(anchor)
    1.0 #uw8(pet therapy).(title)
    3.0 #uw8(pet therapy).(body)
    1.0 #uw8(pet therapy).(heading))))
```



# Galago Query Language

73

- A document is viewed as a sequence of text that may contain arbitrary tags.
  - HTML tags, XML tags
- A single *context* is generated for each unique tag name  $T$ .
  - All text and tags that appear within tags of type  $T$ .
  - Examples: `<body>`, `<title>`, `<h1>`, ...
  - Context may be nested
  - Terms can appear in multiple contexts.
  - Tags used beyond mere structure: Entity / feature extraction
- An *extent* is a sequence of text that appears within a single begin/end tag pair of the same type as the context.

# Galago Query Language

74

```
<html>
<head>
<title>Department Descriptions</title>
</head>
<body>
The following list describes ...
<h1>Agriculture</h1> ...
<h1>Chemistry</h1> ...
<h1>Computer Science</h1> ...
<h1>Electrical Engineering</h1> ...
</body>
</html>
```

```
title context:
<title>Department Descriptions</title>
```

```
h1 context:
<h1>Agriculture</h1>
<h1>Chemistry</h1> ...
<h1>Computer Science</h1> ...
<h1>Electrical Engineering</h1> ...
```

```
body context:
<body> The following list describes ...
<h1>Agriculture</h1> ...
<h1>Chemistry</h1> ...
<h1>Computer Science</h1> ...
<h1>Electrical Engineering</h1> ...
</body>
```

# Galago Query Language – Terms

75

- Term is basic building block.
  - Corresponds to representation nodes in inference network
- Large variety of terms defined
  - Simple, ordered phrase, synonym, ...
- Simple terms:
  - **term**
    - ◇ term that will be normalized and stemmed.
  - **"term"**
    - ◇ term is not normalized or stemmed.
  - Examples:
    - ◇ **presidents**
    - ◇ **"NASA"**

# Galago Query Language – Proximity Terms

76

- **#N( ... )**
  - Ordered window – terms must appear ordered, with at most N-1 terms between each.
- **#od( ... )**
  - Unlimited ordered window – all terms must appear ordered anywhere within current context.
- **#uwN( ... )**
  - Unordered window – all terms must appear within a window of length N in any order.
- **#uw( ... )**
  - Unlimited unordered window – all terms must appear within current context in any order.
- **Examples:**
  - **#1(white house)** – matches “white house” as an exact phrase.
  - **#2(white house)** – matches “white \* house” (where \* is any word or null).
  - **#uw2(white house)** – matches “white house” and “house white”.

# Galago Query Language – Synonyms

77

- **#syn( ... )**
  - Treat all listed terms as synonyms
- **#wsyn( ... )**
  - Treat all listed terms as synonyms
  - Allows assignment of weights
- **Examples:**
  - **#syn(dog canine)** – simple synonym based on two terms.
  - **#syn( #1(united states) #1(united states of america))**  
– creates a synonym from two proximity terms.
  - **#wsyn( 1.0 donald 0.8 don 0.5 donnie )** – weighted synonym indicating relative importance of terms.

# Galago Query Language – Anonymous Terms

78

- **#any( . )**
  - Used to match extent types
- **Examples:**
  - **#any(PERSON)** – matches any occurrence of a person extent.
  - **#1(lincoln died in #any(DATE))** – matches exact phrases of the form: “lincoln died in *<date>...</date>*”.

# Galago Query Language – Context Restriction and Evaluation

79

- **expression.C1,...,CN**
  - Matches when the expression appears in all contexts C1 through CN.
- **expression.(C1,...,CN)**
  - Evaluates the expression using the language model defined by the concatenation of contexts C1...CN within the document.
- Examples:
  - **dog.title** – matches the term “dog” appearing in a title extent.
  - **#uw(smith jones).author** – matches when the two names “smith” and “jones” appear in an author extent.
  - **dog.(title)** – evaluates the term based on the title language model for the document: Probability of occurrence for dog based on number of times word occurs in title field, normalized for number of words in title. Smoothing using only title fields in collection
  - **#1(abraham lincoln).person.(header)** – builds a language model from all of the “header” text in the document and evaluates #1(abraham lincoln).person in that context (i.e. matches only the exact phrase appearing within a person extent within the header context).

# Galago Query Language – Belief Operators

80

- Used to combine evidence
- Weights can specify relative importance of evidence.
- **#combine(...)**
  - Normalized version of the  $bel_{and}(q)$  operator in the inference network model.
- **#weight(...)**
  - Normalized version of the  $bel_{wand}(q)$  operator.
- **#filter(...)**
  - Similar to **#combine**, but with the difference that all terms (simple, proximity, synonym, etc.) are evaluated *without* smoothing. Document must contain at least one instance of the term.



# Galago Query Language – Belief Operators

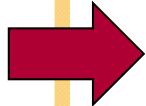
81

- `#combine( <dog canine> training )`
  - Rank by two terms, one of which is a synonym.
- `#combine( biography <#1(president lincoln) #1(abraham lincoln)> )`
  - Rank using two terms, one of which is a synonym of “president lincoln” and “abraham lincoln”.
- `#weight( 1.0 #1(civil war) 3.0 lincoln 2.0 speech )`
  - Rank using three terms, and weight the term “lincoln” as most important, followed by “speech”, then “civil war”.
- `#filter( aquarium #combine(tropical fish) )`
  - Consider only those documents containing the word “aquarium” and rank them according to the query `#combine(tropical fish)`.
- `#filter( #weight( 2.0 europe 1.0 travel )  
#1(john smith).author )`
  - Rank documents about “europe” and “travel” that have “John Smith” in the author context.

# Overview

82

- Older models
- Probabilistic models
- Language models
- Combining evidence
- Web search
- Learning to Rank



# Web Search

83

- Retrieval models in practice
  - Web search most important, but not only, search application
- Major differences to TREC news
  - Size of collection
    - ◇ Billions
  - Connections between documents
    - ◇ Links
  - Range of document types
  - Importance of spam
  - Volume of queries
    - ◇ Tens of millions per day
  - Range of query types
    - ◇ Informational navigational, transactional

# Search Taxonomy

84

## ■ *Informational*

- Finding information about some topic which may be on one or more web pages
- Topical search

## ■ *Navigational*

- Finding a particular web page that the user has either seen before or is assumed to exist
- Known-item search

## ■ *Transactional*

- Finding a site where a task such as shopping or downloading music can be performed

# Web Search

85

- For effective navigational and transactional search, need to combine features that reflect *user relevance*.
- Commercial web search engines combine evidence from *hundreds* of features to generate a ranking score for a web page
  - Page content
  - Page metadata
    - ◇ “age”, how often it is updated
    - ◇ URL of the page
    - ◇ Domain name of its site
    - ◇ Amount of text content
  - Anchor text
  - Links (e.g., PageRank)
  - User behavior (click logs)

# Search Engine Optimization

86

- *SEO*: Understanding the relative importance of features used in search and how they can be manipulated to obtain better search rankings for a web page
  - Improve the text used in the title tag
  - Improve the text in heading tags
  - Make sure that the domain name and URL contain important keywords
  - Improve the anchor text and link structure
- Some of these techniques are regarded as not appropriate by search engine companies

# Web Search

87

- In TREC evaluations, most effective features for navigational search are:
  - Text in the title, body, headings (h1, h2, h3, and h4)
  - Anchor text of all links pointing to the document
  - PageRank number and inlink count
- Given size of Web, many pages will contain all query terms
  - Search engines can use AND semantics
    - ◇ Dangerous for smaller collections
      - Site search, news search, ...
      - TREC: Only 50% of relevant pages contain all search terms
  - Ranking algorithm focuses on discriminating between these pages
  - Word proximity is important

# Term Proximity

88

- Assumption: Query terms are likely to appear in close proximity within relevant documents
  - “Green party political views”
- Many models have been developed
  - N-grams are commonly used in commercial web search
- Dependence model based on inference net has been effective in TREC - e.g.
- Let  $S_Q$  be the set of all non-empty subsets of  $Q$  (power set)
  - Every  $s \in S_Q$  that consists of contiguous query terms is likely to appear as an exact phrase in a relevant document
    - ◇ Represented using the #1 operator
  - Every  $s \in S_Q$  such that  $|s| > 1$  is likely to appear (ordered or unordered) within a reasonably sized window of text in a relevant document
    - ◇ Represented as #uw8 for  $|s| = 2$  and #uw12 for  $|s| = 3$



# Term Proximity

89

- Example query „embryonic stem cells“
- Compiled to Galago query
  - `#weight(`
    - `0.8 #combine( embryonic stem cells )`
    - `0.1 #combine( #od1(stem cells)`
      - `#od1(embryonic stem)`
      - `#od1(embryonic stem cells) )`
    - `0.1 #combine( #uw8(stem cells)`
      - `#uw8(embryonic cells)`
      - `#uw8(embryonic stem)`
      - `#uw12(embryonic stem cells) )`
  - `)`

# Example Web Query

90

- Query: „pet therapy“
- Compiled to Galago query
- #weight(
 

```

0.1 #weight( 0.6 #prior(pagerank) 0.4 #prior(inlinks))
1.0 #weight(
  0.9 #combine(
    #weight(1.0 pet.(anchor) 1.0 pet.(title)
      3.0 pet.(body) 1.0 pet.(heading))
    #weight(1.0 therapy.(anchor) 1.0 therapy.(title)
      3.0 therapy.(body) 1.0 therapy.(heading)))
0.1 #weight(
  1.0 #od1(pet therapy).(anchor)
  1.0 #od1(pet therapy).(title)
  3.0 #od1(pet therapy).(body)
  1.0 #od1(pet therapy).(heading))
0.1 #weight(
  1.0 #uw8(pet therapy).(anchor)
  1.0 #uw8(pet therapy).(title)
  3.0 #uw8(pet therapy).(body)
  1.0 #uw8(pet therapy).(heading))))

```

PageRank and inlinks calculated at index time

Proximity can be index, but increases index size

# Query types

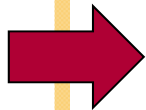
91

- Insights gained from TREC experiments
- Topical search:
  - Simple terms and proximity features suffice
- Navigational search:
  - More evidence is helpful
- Pseudo-relevance feedback
  - Helps topical search
  - Is detrimental for navigational search
- But: How can we determine query type?
- Other evidence is in general useful
  - User behavior: Clicked-on pages, dwell time, links followed
- But: How to weight and combine more and more evidence?
  - Idea: Machine learning

# Overview

92

- Older models
- Probabilistic models
- Language models
- Combining evidence
- Web search
- Learning to Rank



# Machine Learning and IR

93

- Considerable interaction between these fields
  - Rocchio algorithm (60s) is a simple learning approach
  - 80s, 90s: learning ranking algorithms based on user feedback
  - 2000s: text categorization
- Limited by amount of training data
- Web query logs have generated new wave of research
  - e.g., “Learning to Rank”

# Generative vs. Discriminative

94

- All of the probabilistic retrieval models presented so far fall into the category of *generative models*.
  - A generative model assumes that documents were generated from some underlying model (in this case, usually a multinomial distribution) and uses training data to estimate the parameters of the model.
  - Probability of belonging to a class (i.e. the relevant documents for a query) is then estimated using Bayes' Rule and the document model.

# Generative vs. Discriminative

95

- A *discriminative* model estimates the probability of belonging to a class directly from the observed features of the document based on the training data.
- Generative models perform well with low numbers of training examples.
- Discriminative models usually have the advantage given enough training data.
  - Can also easily incorporate many features

# Discriminative Models for IR

96

- Discriminative models can be trained using explicit relevance judgments or click data in query logs
  - Click data is much cheaper, more noisy
  - e.g. Ranking Support Vector Machine (SVM) takes as input *partial rank* information for queries
    - ◇ Partial information about which documents should be ranked higher than others



# Summary

107

- Best retrieval model depends on application and data available
- Evaluation corpus (or test collection), training data, and user data are all critical resources.
- Language resources (e.g., thesaurus) can make a big difference