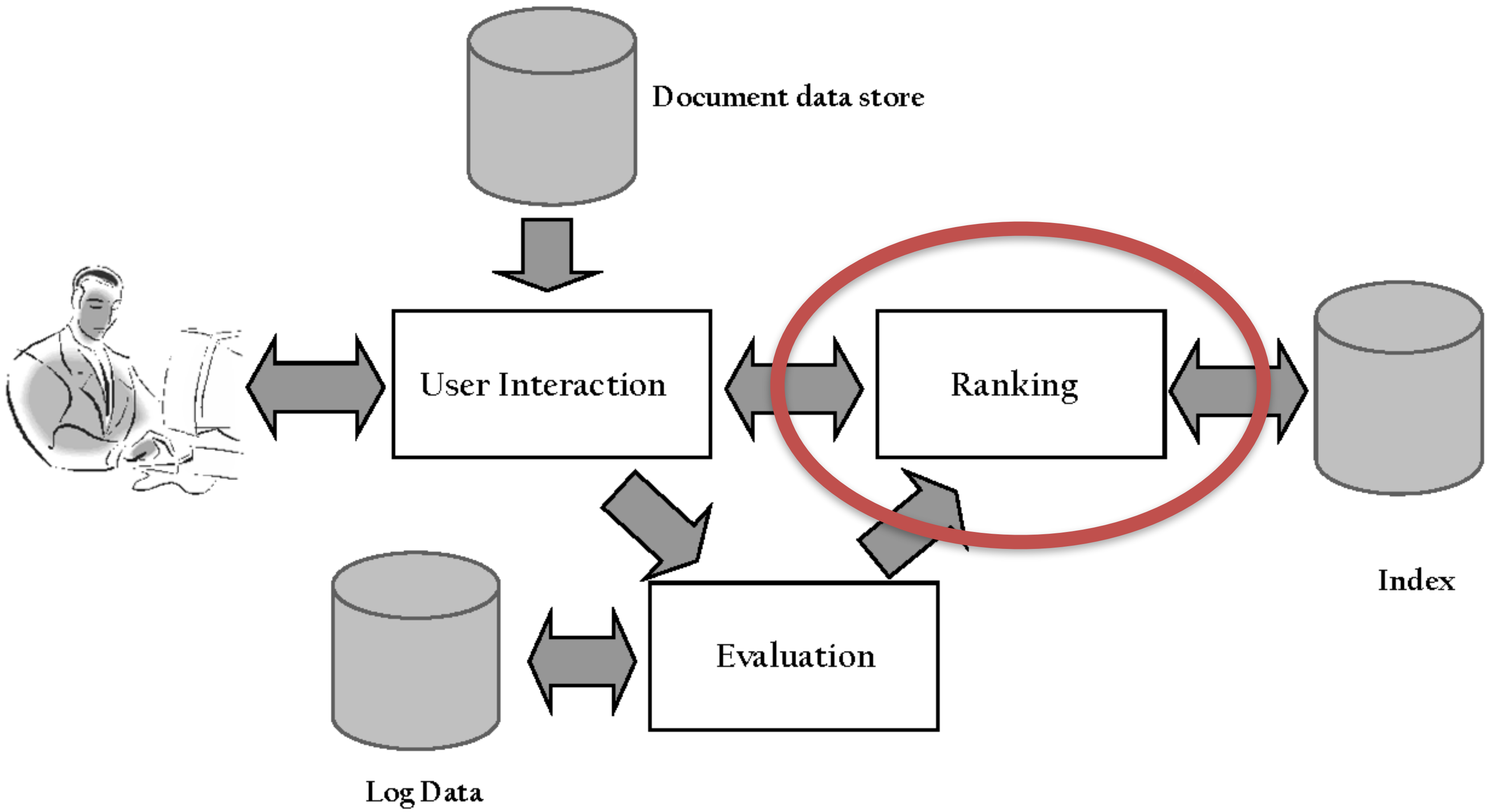
CS6200
Information Retrieval

David Smith
College of Computer and Information Science
Northeastern University

Query Process



Retrieval Models

- Provide a mathematical framework for defining the search process
  - includes explanation of assumptions
  - basis of many ranking algorithms
  - can be implicit
    - Retrieval model developed by trial and error
- Progress in retrieval models has corresponded with improvements in effectiveness
- Theories about—i.e., models of—relevance

Relevance

- Complex concept that has been studied for some time
  - Many factors to consider
  - People often disagree when making relevance judgments
- Retrieval models make various assumptions about relevance to simplify problem
  - e.g., *topical* vs. *user* relevance
  - e.g., *binary* vs. *multi-valued* relevance

Topical vs. User Relevance

- Topical Relevance
  - Document and query are on the same topic
  - Query: “U.S. Presidents”
  - Document: Wikipedia article on Abraham Lincoln
- User Relevance
  - Incorporate factors beside document topic
    - Document freshness
    - Style
    - Content presentation

Binary vs. Multi-Valued Relevance

- Binary Relevance
  - The document is either relevant or not
- Multi-Valued Relevance
  - Makes the evaluation task easier for the judges
  - Not as important for retrieval models
  - Many retrieval models calculate the *probability of relevance*

Retrieval Model Overview

- Older models
  - Boolean retrieval
  - Vector Space model
- Probabilistic Models
  - BM25
  - Language models
- Combining evidence
  - Inference networks
  - Learning to Rank

Boolean Retrieval

- Two possible outcomes for query processing
  - TRUE and FALSE
  - “exact-match” retrieval; “set” retrieval
  - simplest form of ranking
- Query usually specified using Boolean operators
  - AND, OR, NOT
  - proximity operators and wildcards also used

Boolean Retrieval

- Advantages
  - Results are predictable, relatively easy to explain
  - Many different features can be incorporated
  - Efficient processing since many documents can be eliminated from search
- Disadvantages
  - Effectiveness depends entirely on user
  - Simple queries usually don’t work well
  - Complex queries are difficult

Searching by Numbers

- Sequence of queries driven by number of retrieved documents
  1. lincoln
  2. president AND lincoln
  3. president AND lincoln AND NOT (automobile OR car)
  4. president AND lincoln AND biography AND life AND birthplace AND gettysburg AND NOT (automobile OR car)
  5. president AND lincoln AND (biography OR life OR birthplace OR gettysburg) AND NOT (automobile OR car)

Vector Space Model

- Documents and query represented by a vector of term weights
- Collection represented by a matrix of term weights

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

$$\begin{array}{ccccc}
 & Term_1 & Term_2 & \ldots & Term_t \\
Doc_1 & d_{11} & d_{12} & \ldots & d_{1t} \\
Doc_2 & d_{21} & d_{22} & \ldots & d_{2t} \\
\vdots & \vdots & & & \\
Doc_n & d_{n1} & d_{n2} & \ldots & d_{nt}
\end{array}$$

Vector Space Model

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

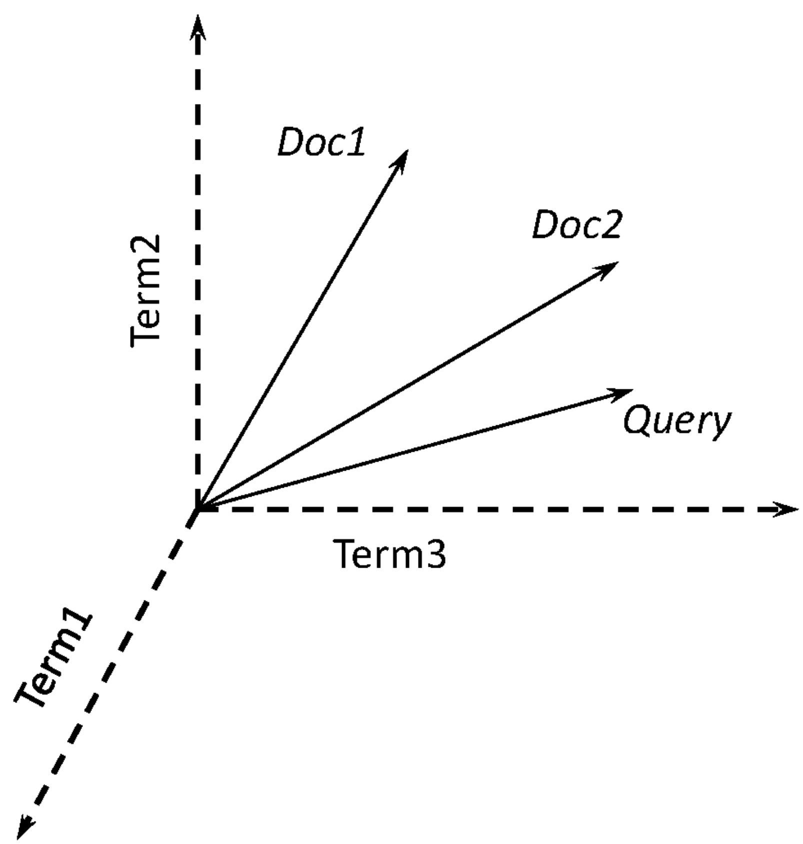
Vector Space Model

- Query: “tropical fish”

Term	Query
aquarium	0
bowl	0
care	0
fish	**1**
freshwater	0
goldfish	0
homepage	0
keep	0
setup	0
tank	0
tropical	**1**

Vector Space Model

- 3-d pictures useful, but can be misleading for high-dimensional space



Vector Space Model

- Documents ranked by distance between points representing query and documents
  - *Similarity* measure more common than a distance or *dissimilarity* measure
  - e.g. Cosine correlation

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

Similarity Calculation

– 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), Q = (1.5,

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

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

Difference from Boolean Retrieval

- Similarity calculation has two factors that distinguish it from Boolean retrieval
  - Number of matching terms affects similarity
  - Weight of matching terms affects similarity
- Documents can be *ranked* by their similarity scores

Term Weights

- *tf.idf* weight
  - Term frequency weight measures importance in document: $tf_{ik} = \frac{f_{ik}}{\sum_{j=1}^{t} f_{ij}}$
  - Inverse document frequency measures importance in collection: $idf_k = \log \frac{N}{n_k}$
  - Heuristic combination

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

Relevance Feedback

- Rocchio algorithm
- *Optimal query*
  - Maximizes the difference between the average vector representing the relevant documents and the average vector representing the non-relevant documents
- Modifies query according to

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

  - α, β, and γ are parameters
    - Typical values 8, 16, 4

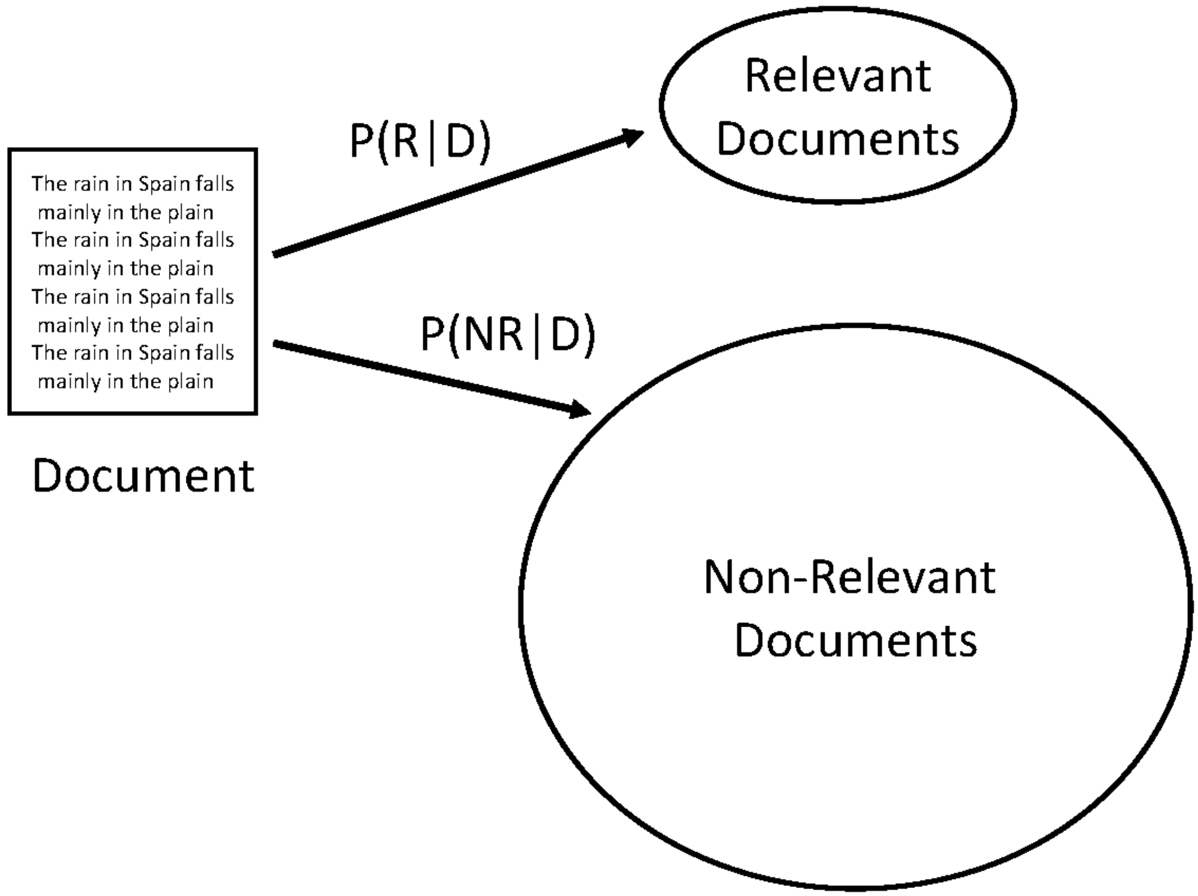
Vector Space Model

- Advantages
  - Simple computational framework for ranking
  - Any similarity measure or term weighting scheme could be used
- Disadvantages
  - Assumption of term independence
  - No *predictions* about techniques for effective ranking

Probability Ranking Principle

- 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.”

IR as Classification



Bayes Classifier

- Bayes Decision Rule
  - A document D is relevant if $P(R|D) > P(NR|D)$
- Estimating probabilities
  - use Bayes Rule

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

  - classify a document as relevant if

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

    - This is *likelihood ratio*

Estimating P(D|R)

- Assume independence

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

- *Binary independence model*
  - document represented by a vector of binary features indicating term occurrence (or non-occurrence)
  - 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

Binary Independence Model

$$\frac{P(D|R)}{P(D|NR)} = \prod_{i:d_i=1} \frac{p_i}{s_i} \cdot \prod_{i:d_i=0} \frac{1-p_i}{1-s_i}$$

$$= \prod_{i:d_i=1} \frac{p_i}{s_i} \cdot (\prod_{i:d_i=1} \frac{1-s_i}{1-p_i} \cdot \prod_{i:d_i=1} \frac{1-p_i}{1-s_i}) \cdot \prod_{i:d_i=0} \frac{1-p_i}{1-s_i}$$

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

Binary Independence Model

- 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
- If we assume p_i constant, s_i approximated by entire collection, get *idf*-like weight

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

Contingency Table

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

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

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

Gives scoring function:

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

BM25

- Popular and effective ranking algorithm based on binary independence model
  - adds document and query term weights

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

  - k_1, k_2 and K are parameters whose values are set empirically
  - $K = k_1((1-b) + b \cdot \frac{dl}{avdl})$ *dl* is doc length
  - Typical TREC value for k_1 is 1.2, k_2 varies from 0 to 1000, b = 0.75

BM25 Example

- Query with two terms, "president lincoln", (*qf* = *1)*
- No relevance information (*r and R are* zero)
- 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 ($dl/avdl$ = .9)
- k_1 = 1.2, b = 0.75, and k_2 = 100
- $K = 1.2 \cdot (0.25 + 0.75 \cdot 0.9) = 1.11$

BM25 Example

$$
\begin{aligned}
BM25(Q,D) \quad = \quad & \\
& \log \frac{(0+0.5)/(0-0+0.5)}{(40000-0+0.5)/(500000-40000-0+0+0.5)} \\
& \times \frac{(1.2+1)15}{1.11+15} \times \frac{(100+1)1}{100+1} \\
& + \log \frac{(0+0.5)/(0-0+0.5)}{(300-0+0.5)/(500000-300-0+0+0.5)} \\
& \times \frac{(1.2+1)25}{1.11+25} \times \frac{(100+1)1}{100+1} \\
\\
= \quad & \log 460000.5/40000.5 \cdot 33/16.11 \cdot 101/101 \\
& + \log 499700.5/300.5 \cdot 55/26.11 \cdot 101/101 \\
= \quad & 2.44 \cdot 2.05 \cdot 1 + 7.42 \cdot 2.11 \cdot 1 \\
= \quad & 5.00 + 15.66 = 20.66
\end{aligned}
$$

BM25 Example

- 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

Language Model

- Language model
  - Probability distribution over strings of text
- **Unigram** language model
  - generation of text consists of pulling words out of a “bucket” according to the probability distribution and replacing them
- **N-gram** language model
  - some applications use bigram and trigram language models where probabilities depend on previous words

Language Model

- A *topic* in a document or query can be represented as a language model
  - i.e., words that tend to occur often when discussing a topic will have high probabilities in the corresponding language model
- *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
  - doesn’t model *burstiness*

LMs for Retrieval

- 3 possibilities:
  - probability of generating the query text from a document language model
  - probability of generating the document text from a query language model
  - comparing the language models representing the query and document topics
- Models of topical relevance

Query-Likelihood Model

- Rank documents by the probability that the query could be generated by the document model (i.e. same topic)
- Given query, start with P(D|Q)
- Using Bayes' Rule

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

- Assuming prior is uniform, unigram model

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

Estimating Probabilities

- 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
- If query words are missing from document, score will be zero
  – Missing 1 out of 4 query words same as missing 3 out of 4

Smoothing

- 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
  - What does this do to the likelihood of the document?

Estimating Probabilities

- 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
- Estimate for words that occur is

$$(1 - a_D)\, P(q_i | D) + a_D P(q_i | C)$$

- Different forms of estimation come from different a_D

Jelinek-Mercer Smoothing

- a_D is a constant, λ
- Gives estimate of

$$p(q_i|D) = (1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|}$$

- Ranking score

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

- Use logs for convenience
  - accuracy problems multiplying small numbers

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

Where is *tf.idf* Weight?

$$
\begin{aligned}
\log P(Q|D) &= \sum_{i=1}^{n} \log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|}) \\
&= \sum_{i:f_{q_i,D}>0} \log((1-\lambda)\frac{f_{q_i,D}}{|D|} + \lambda\frac{c_{q_i}}{|C|}) + \sum_{i:f_{q_i,D}=0} \log(\lambda\frac{c_{q_i}}{|C|})
\end{aligned}
$$

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

- proportional to the term frequency, inversely proportional to the collection frequency

Dirichlet Smoothing

- a_D depends on document length

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

- Gives probability estimation of

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

- and document score

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

Query Likelihood Example

- 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
- $\mu = 2{,}000$

Query Likelihood Example

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

- Negative number because summing logs of small numbers

Query Likelihood Example

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