Information Retrieval (IR)
Result Ranking

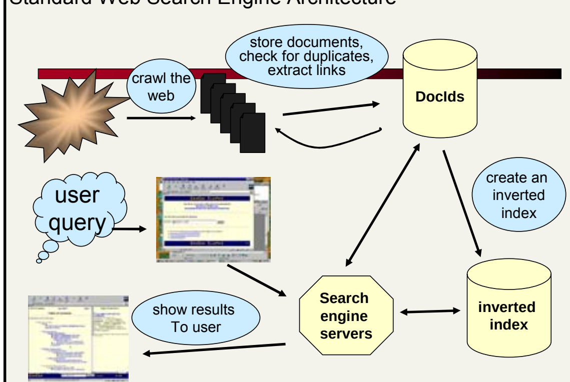
CSE 454

Class Overview

- Other Cool Stuff
- Query processing
- Indexing
- **IR - Ranking**
- Content Analysis
- Crawling
- Document Layer
- Network Layer

A Closeup View

Standard Web Search Engine Architecture



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

Retrieval Model Overview

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

Test Corpora

TABLE 4.3 Common Test Corpora

Collection	*NDocs*	*NQrys*	*Size (MB)*	*Term/Doc*	*Q-D RelAss*
ADI	82	35			
AIT	2109	14	2	400	>10,000
CACM	3204	64	2	24.5	
CISI	1460	112	2	46.5	
Cranfield	1400	225	2	53.1	
LISA	5872	35	3		
Medline	1033	30	1		
NPL	11,429	93	3		
OSHMED	34,8566	106	400	250	16,140
Reuters	21,578	672	28	131	
TREC	740,000	200	2000	89-3543	» 100,000

slide from Raghavan, Schütze, Larson

Standard relevance benchmarks

- TREC - National Institute of Standards and Testing (NIST) has run large IR testbed for many years
- Reuters and other benchmark sets used
- "Retrieval tasks" specified
  - sometimes as queries
- Human experts mark, for each query and for each doc, "Relevant" or "Not relevant"
  - or at least for subset that some system returned

slide from Raghavan, Schütze, Larson

Precision and recall

- **Precision**: fraction of retrieved docs that are relevant = P(relevant|retrieved)
- **Recall**: fraction of relevant docs that are retrieved = P(retrieved|relevant)

	Relevant	Not Relevant
Retrieved	tp	fp
Not Retrieved	fn	tn

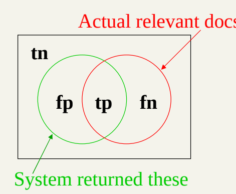
- Precision P = tp/(tp + fp)
- Recall R = tp/(tp + fn)

slide from Raghavan, Schütze, Larson

Precision & Recall

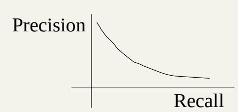
Precision $\frac{tp}{tp+fp}$

Proportion of selected items that are correct



Recall $\frac{tp}{tp+fn}$

% of target items that were selected



Precision-Recall curve
Shows tradeoff

Precision/Recall

- Can get high recall (but low precision)
  - Retrieve all docs on all queries!
- Recall is a non-decreasing function of the number of docs retrieved
  - Precision usually decreases (in a good system)
- Difficulties in using precision/recall
  - Binary relevance
  - Should average over large corpus/query ensembles
  - Need human relevance judgements
  - Heavily skewed by corpus/authorship

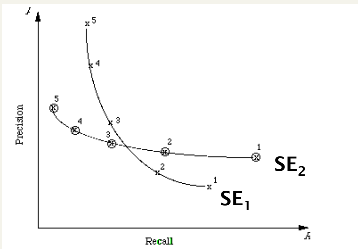
slide from Raghavan, Schütze, Larson

Precision-recall curves

- Evaluation of ranked results:
  - You can return any number of results ordered by similarity
  - By taking various numbers of documents (levels of recall), you can produce a *precision-recall curve*

slide from Raghavan, Schütze, Larson

Precision-recall curves



slide from Raghavan, Schütze, Larson

A combined measure: F

- Combined measure that assesses this tradeoff is F measure (weighted harmonic mean):

$$F = \frac{1}{\alpha\frac{1}{P} + (1-\alpha)\frac{1}{R}} = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

- People usually use balanced F_1 measure
  - i.e., with $\beta = 1$ or $\alpha = ½$
- Harmonic mean is conservative average
  - See CJ van Rijsbergen, *Information Retrieval*

slide from Raghavan, Schütze, Larson

Evaluation

- There are various other measures
  - Precision at fixed recall
    - This is perhaps the most appropriate thing for web search: all people want to know is how many good matches there are in the first one or two pages of results
  - 11-point interpolated average precision
    - The standard measure in the TREC competitions: Take the precision at 11 levels of recall varying from 0 to 1 by tenths of the documents, using interpolation (the value for 0 is always interpolated!), and average them

slide from Raghavan, Schütze, Larson

Boolean Retrieval

- **Two possible outcomes for query processing**
  - TRUE and FALSE
  - "exact-match" retrieval
  - simplest form of ranking
- **Query specified w/ Boolean operators**
  - AND, OR, NOT
  - proximity operators also used

from Croft, Metzler, Strohman. © Addison Wesley

Query

- Which plays of Shakespeare contain the words ***Brutus*** *AND* ***Caesar*** but *NOT* ***Calpurnia***?

slide from Raghavan, Schütze, Larson

Term-document incidence

	Tempest	Hamlet	Othello	Macbeth
Antony	0	0	0	1
Brutus	0	1	0	0
Caesar	0	1	1	1
Calpurnia	0	0	0	0
Cleopatra	0	0	0	0
mercy	1	1	1	1
worser	1	1	1	0

1 if **play** contains **word**,
0 otherwise

slide from Raghavan, Schütze, Larson

Booleans over Incidence Vectors

- So we have a 0/1 vector for each term.
- To answer query: take the vectors for ***Brutus, Caesar*** and ***Calpurnia*** (complemented) ➔ bitwise *AND*.
- 110100 *AND* 110111 *AND* 101111 = 100100.

slide from Raghavan, Schütze, Larson

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

from Croft, Metzler, Strohman. © Addison Wesley

Interlude

- Better Models Coming Soon:
  - Vector Space model
  - Probabilistic Models
    - BM25
    - Language models
- Shared Issues – What to Index
  - Punctuation
  - Case Folding
  - Stemming
  - Stop Words
  - Spelling
  - Numbers

Issues in what to index

Cooper's concordance of Wordsworth was published in 1911. The applications of full-text retrieval are legion: they include résumé scanning, litigation support and searching published journals on-line.

- ***Cooper's*** vs. ***Cooper*** vs. ***Coopers***.
- ***Full-text*** vs. ***full text*** vs. {***full, text***} vs. ***fulltext.***
- Accents: ***résumé*** vs. ***resume***.

slide from Raghavan, Schütze, Larson

Punctuation

- ***Ne'er***: use language-specific, handcrafted "locale" to normalize.
- ***State-of-the-art***: break up hyphenated sequence.
- ***U.S.A.*** vs. ***USA*** - use locale.
- ***a.out***

slide from Raghavan, Schütze, Larson

Numbers

- 3/12/91
- Mar. 12, 1991
- 55 B.C.
- B-52
- 100.2.86.144
  - Generally, don't index as text
  - Creation dates for docs

slide from Raghavan, Schütze, Larson

Case folding

- Reduce all letters to lower case
- Exception: upper case in mid-sentence
  - *e.g.,* ***General Motors***
  - ***Fed*** vs. ***fed***
  - ***SAIL*** vs. ***sail***

slide from Raghavan, Schütze, Larson

Thesauri and soundex

- Handle synonyms and homonyms
  - Hand-constructed equivalence classes
    - e.g., ***car*** = ***automobile***
    - ***your*** ≠ ***you're***
- Index such equivalences?
- Or expand query?

slide from Raghavan, Schütze, Larson

Spell correction

- Look for all words within (say) edit distance 3 (Insert/Delete/Replace) at query time
  - *e.g.,* ***Alanis Morisette***
- Spell correction is expensive and slows the query (up to a factor of 100)
  - Invoke only when index returns zero matches?
  - What if docs contain mis-spellings?

slide from Raghavan, Schütze, Larson

Lemmatization

- Reduce inflectional/variant forms to base form
- E.g.,
  - *am, are, is* → *be*
  - *car, cars, car's, cars'* → *car*
- *the boy's cars are different colors* → *the boy car be different color*

slide from Raghavan, Schütze, Larson

Stemming

See hidden slides

- Reduce terms to their "roots" before indexing
  - language dependent
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

for example compressed and compression are both accepted as equivalent to compress.	➡	for exampl compres and compres are both accept as equival to compres.

slide from Raghavan, Schütze, Larson

Porter's algorithm

- Most common algorithm for stemming English
- Conventions + 5 phases of reductions
  - phases applied sequentially
  - each phase consists of a set of commands
  - sample convention: *Of the rules in a compound command, select the one that applies to the longest suffix.*
- Porter's stemmer available: http//www.sims.berkeley.edu/~hearst/irbook/porter.html

slide from Raghavan, Schütze, Larson

Typical rules in Porter

- *sses* → *ss*
- *ies* → *i*
- *ational* → *ate*
- *tional* → *tion*

slide from Raghavan, Schütze, Larson

Challenges

- Sandy
- Sanded

➔ Sand ???

slide from Raghavan, Schütze, Larson

Beyond Term Search

- Phrases?
- Proximity: Find ***Gates*** *NEAR* ***Microsoft***.
  - Index must capture position info in docs.
- Zones in documents: Find documents with (*author* = ***Ullman***) *AND* (text contains ***automata***).

slide from Raghavan, Schütze, Larson

Ranking search results

- Boolean queries give inclusion or exclusion of docs.
- Need to measure proximity from query to each doc.
- Whether docs presented to user are singletons, or a group of docs covering various aspects of the query.

slide from Raghavan, Schütze, Larson

Ranking models in IR

- Key idea:
  - *We wish to return in order the documents most likely to be useful to the searcher*
- To do this, we want to know which documents *best* satisfy a query
  - An obvious idea is that if a document talks about a topic *more* then it is a better match
- A query should then just specify terms that are relevant to the information need, without requiring that all of them must be present
  - Document relevant if it has a lot of the terms

slide from Raghavan, Schütze, Larson

Retrieval Model Overview

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

from Croft, Metzler, Strohman, © Addison Wesley

Binary term presence matrices

- Record whether a document contains a word: document is binary vector in $\{0,1\}^v$
- Idea: Query satisfaction = overlap measure:

$$|X \cap Y|$$

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth
Antony	1	1	0	0	0	1
Brutus	1	1	0	1	0	0
Caesar	1	1	0	1	1	1
Calpurnia	0	1	0	0	0	0
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0

slide from Raghavan, Schütze, Larson

Overlap matching

- What are the problems with the overlap measure?
- It doesn't consider:
  - Term frequency in document
  - Term scarcity in collection
    - (How many documents mention term?)
  - Length of documents

slide from Raghavan, Schütze, Larson

Many Overlap Measures

$\lvert Q \cap D \rvert$	Simple matching (coordination level match)
$2\dfrac{\lvert Q \cap D \rvert}{\lvert Q \rvert + \lvert D \rvert}$	Dice's Coefficient
$\dfrac{\lvert Q \cap D \rvert}{\lvert Q \cup D \rvert}$	Jaccard's Coefficient
$\dfrac{\lvert Q \cap D \rvert}{\lvert Q \rvert^{1/2} \times \lvert D \rvert^{1/2}}$	Cosine Coefficient
$\dfrac{\lvert Q \cap D \rvert}{\min(\lvert Q \rvert, \lvert D \rvert)}$	Overlap Coefficient

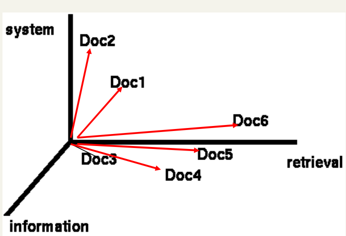
slide from Raghavan, Schütze, Larson

Documents as vectors

- Each doc *j* can be viewed as a vector of *tf* values, one component for each term
- So we have a vector space
  - terms are axes
  - docs live in this space
  - even with stemming, may have 20,000+ dimensions
- (The corpus of documents gives us a matrix, which we could also view as a vector space in which words live – transposable data)

slide from Raghavan, Schütze, Larson

Documents in 3D Space



Assumption: Documents that are "close together" in space are similar in meaning.

slide from Raghavan, Schütze, Larson

The vector space model

Query as vector:

- Regard query as *short document*
- Return the docs, ranked by distance to the query
- Easy to compute, since both query & docs are vectors.
- Developed in the SMART system (Salton, c. 1970) and standardly used by TREC participants and web IR systems

slide from Raghavan, Schütze, Larson

Vector Representation

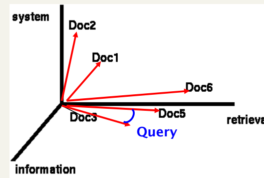
- Documents & Queries represented as **vectors**.
- Position 1 corresponds to term 1, …position t to term t
- The **weight** of the term is stored in each position

$$D_i = w_{d_{i1}}, w_{d_{i2}}, \ldots, w_{d_{it}}$$
$$Q = w_{q1}, w_{q2}, \ldots, w_{qt}$$
$$w = 0 \text{ if a term is absent}$$

- Vector distance measure used to rank retrieved documents

slide from Raghavan, Schütze, Larson

Documents in 3D Space



Documents that are close to query
(measured using vector-space metric)
=> returned first.

slide from Raghavan, Schütze, Larson

Document Space has High Dimensionality

- What happens beyond 2 or 3 dimensions?
  - Similarity still has to do with the number of shared tokens.
  - More terms -> harder to understand which subsets of words are shared among similar documents.
- We will look in detail at ranking methods
  - One approach to handling high dimensionality: Clustering

slide from Raghavan, Schütze, Larson

Word Frequency

- Which word is more indicative of document similarity?
  - 'book,' or 'Rumplestiltskin'?
  - Need to consider "document frequency"---how frequently the word appears in doc collection.
- Which doc is a better match for the query "Kangaroo"?
  - One with a single mention of Kangaroos… or a doc that mentions it 10 times?
  - Need to consider "term frequency"---how many times the word appears in the current document.

slide from Raghavan, Schütze, Larson

TF x IDF

$$w_{ik} = tf_{ik} * \log(N / n_k)$$

T_k = term k in document D_i

tf_{ik} = frequency of term T_k in document D_i

idf_k = inverse document frequency of term T_k in C

$$idf_k = \log\left(\frac{N}{n_k}\right)$$

N = total number of documents in the collection C

n_k = the number of documents in C that contain T_k

slide from Raghavan, Schütze, Larson

Inverse Document Frequency

- IDF provides high values for rare words and low values for common words

$$\log\left(\frac{10000}{10000}\right) = 0$$
$$\log\left(\frac{10000}{5000}\right) = 0.301$$
$$\log\left(\frac{10000}{20}\right) = 2.698$$
$$\log\left(\frac{10000}{1}\right) = 4$$

slide from Raghavan, Schütze, Larson

TF-IDF normalization

- Normalize the term weights
  - so longer docs not given more weight (fairness)
  - force all values to fall within a certain range: [0, 1]

$$w_{ik} = \frac{tf_{ik} \log(N / n_k)}{\sqrt{\sum_{k=1}^{t} (tf_{ik})^2 [\log(N / n_k)]^2}}$$

slide from Raghavan, Schütze, Larson

Vector space similarity
(use the weights to compare the documents)

Now, the similarity of two documents is:

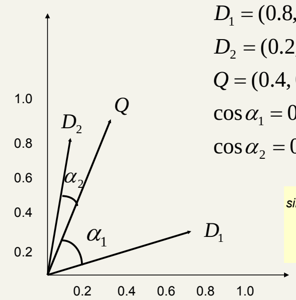
$$sim(D_i, D_j) = \sum_{k=1}^{t} w_{ik} * w_{jk}$$

This is also called the cosine, or normalized inner product.

(Normalization was done when weighting the terms.)

slide from Raghavan, Schütze, Larson

Computing Cosine Similarity Scores



$D_1 = (0.8, 0.3)$

$D_2 = (0.2, 0.7)$

$Q = (0.4, 0.8)$

$\cos \alpha_1 = 0.74$

$\cos \alpha_2 = 0.98$

$$sim(Q, D_2) = \frac{(0.4*0.2)+(0.8*0.7)}{\sqrt{[(0.4)^2+(0.8)^2]*[(0.2)^2+(0.7)^2]}} = \frac{0.64}{\sqrt{0.42}} = 0.98$$

slide from Raghavan, Schütze, Larson

Computing a similarity score

Say we have query vector $Q = (0.4, 0.8)$

Also, document $D_2 = (0.2, 0.7)$

What does their similarity comparison yield?

$$sim(Q, D_2) = \frac{(0.4*0.2)+(0.8*0.7)}{\sqrt{[(0.4)^2+(0.8)^2]*[(0.2)^2+(0.7)^2]}}$$

$$= \frac{0.64}{\sqrt{0.42}} = 0.98$$

slide from Raghavan, Schütze, Larson

To Think About

- How does this ranking algorithm behave?
  - Make a set of hypothetical documents consisting of terms and their weights
  - Create some hypothetical queries
  - How are the documents ranked, depending on the weights of their terms and the queries' terms?

slide from Raghavan, Schütze, Larson

Summary: Why use vector spaces?

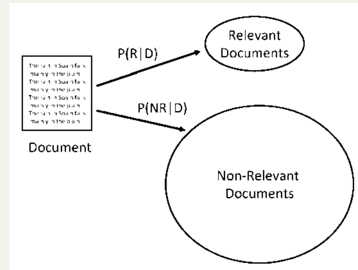
- User's query treated as a (very) short document.
- Query ➔ a vector in the same space as the docs.
- Easily measure each doc's proximity to query.
- Natural measure of scores/ranking
  - No longer Boolean.

slide from Raghavan, Schütze, Larson

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 ant if
    - lhs is $\frac{P(D|R)}{P(D|NR)} > \frac{P(NR)}{P(R)}$

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

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

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

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

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 · (0.25 + 0.75 · 0.9) = 1.11

BM25 Example

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

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

- *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
- 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

Estimating Probabilities

- Estimate for unseen words is $\alpha_D P(q_i|C)$
  - $P(q_i|C)$ is the probability for query word *i* in the *collection* language model for collection *C* (background probability)
  - α_D is a parameter
- Estimate for words that occur is

  $(1 - \alpha_D)\ P(q_i|D) + \alpha_D\ P(q_i|C)$
- Different forms of estimation come from different α_D

Jelinek-Mercer Smoothing

- α_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
  - 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?

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

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

from Croft, Metzler, Strohman. © Addison Wesley

Dirichlet Smoothing

- α_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

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

from Croft, Metzler, Strohman. © Addison Wesley

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
- μ = 2,000

from Croft, Metzler, Strohman. © Addison Wesley

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

from Croft, Metzler, Strohman. © Addison Wesley

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

from Croft, Metzler, Strohman. © Addison Wesley

Relevance Models

- *Relevance model* – language model representing information need
  - query and relevant documents are samples from this 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 difficulties comparing across documents of different lengths

Pseudo-Relevance Feedback

- Estimate relevance model from query and top-ranked documents
- 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

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

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

  - Use negative KL-divergence for ranking, and assume relevance model *R* is the true distribution (not symmetric),

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

KL-Divergence

- Given a simple maximum likelihood estimate for *P(w|R)*, based on the frequency in the query text, ranking score is

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

  - rank-equivalent to query likelihood score
- Query likelihood model is a special case of retrieval based on relevance model

Estimating the Relevance Model

- 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

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

- By definition

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

Estimating the Relevance Model

- Joint probability is

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

- Assume

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

- Gives

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

Estimating the Relevance Model

- *P(D)* usually assumed to be uniform
- *P(w, q1 . . . qn)* 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
- Formal model for pseudo-relevance feedback
  - query expansion technique

Pseudo-Feedback Algorithm

1. Rank documents using the query likelihood score for query Q.
2. Select some number of the top-ranked documents to be the set $\mathcal{C}$.
3. Calculate the relevance model probabilities $P(w|R)$. $P(q_1 \dots q_n)$ is used as a normalizing constant and is calculated 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

$$\sum_w P(w|R) \log P(w|D)$$

Example from Top 10 Docs

president lincoln	*abraham lincoln*	*fishing*	*tropical fish*
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

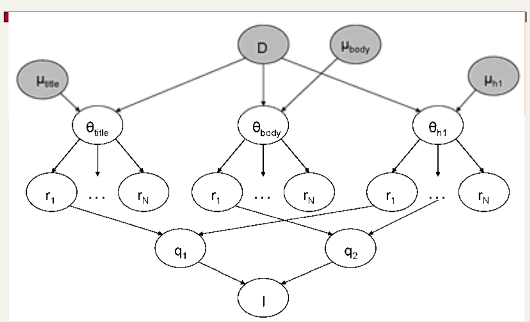
Example from Top 50 Docs

president lincoln	*abraham lincoln*	*fishing*	*tropical fish*
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

Combining Evidence

- Effective retrieval requires the combination of many pieces of evidence about a document's potential relevance
  - have focused on simple word-based evidence
  - many other types of evidence
    - structure, PageRank, metadata, even scores from different models
- *Inference network* model is one approach to combining evidence
  - uses Bayesian network formalism

Inference Network



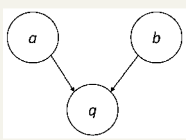
Inference Network

- *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 θ estimated using the parameters μ
  - 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

- *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
- *Information need node* (I) is a special query node that combines all of the evidence from the other query nodes
  - network computes P(I|D, μ)

Example: AND Combination



a and *b* are *parent* nodes for q

$P(q = \text{TRUE}\|a,b)$	a	b
0	FALSE	FALSE
0	FALSE	TRUE
0	TRUE	FALSE
1	TRUE	TRUE

Example: AND Combination

- Combination must consider all possible states of parents
- Some combinations can be computed efficiently

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

Inference Network Operators

$$\begin{aligned} bel_{not}(q) &= 1 - p_1 \\ bel_{or}(q) &= 1 - \prod_i^n (1 - p_i) \\ bel_{and}(q) &= \prod_i^n p_i \\ bel_{wand}(q) &= \prod_i^n p_i^{wt_i} \\ bel_{max}(q) &= max\{p_1, p_2, \ldots, p_n\} \\ bel_{sum}(q) &= \frac{\sum_i^n p_i}{n} \\ bel_{wsum}(q) &= \frac{\sum_i^n wt_i p_i}{\sum_i^n wt_i} \end{aligned}$$

Web Search

- 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, anchor text, links (e.g., PageRank), and user behavior (click logs)
  - page metadata – e.g., “age”, how often it is updated, the URL of the page, the domain name of its site, and the amount of text content

Search Engine Optimization

- *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
  - e.g., 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, and try to improve the anchor text and link structure
  - Some of these techniques are regarded as not appropriate by search engine companies

Web Search

- In TREC evaluations, most effective features for navigational search are:
  - text in the title, body, and heading (h1, h2, h3, and h4) parts of the document, the anchor text of all links pointing to the document, the PageRank number, and the inlink count
- Given size of Web, many pages will contain all query terms
  - Ranking algorithm focuses on discriminating between these pages
  - Word proximity is important

Term Proximity

- 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.

```
#weight(
    0.8 #combine(embryonic stem cells)
    0.1 #combine( #od:1(stem cells) #od:1(embryonic stem)
                  #od:1(embryonic stem cells))
    0.1 #combine( #uw:8(stem cells) #uw:8(embryonic cells)
                  #uw:8(embryonic stem) #uw:12(embryonic stem cells)))
```


Example Web 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 #od:1(pet therapy).(anchor) 1.0 #od:1(pet therapy).(title)
            3.0 #od:1(pet therapy).(body) 1.0 #od:1(pet therapy).(heading))
        0.1 #weight(
            1.0 #uw:8(pet therapy).(anchor) 1.0 #uw:8(pet therapy).(title)
            3.0 #uw:8(pet therapy).(body) 1.0 #uw:8(pet therapy).(heading)))
    )
```


Machine Learning and IR

- 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”

Features

- Page Rank
- Query word in color on page?
- # images on page
- # outlinks on page
- URL length
- Page edit recency

- Page Classifiers (20+)
  - Spam
  - Adult
  - Actor / celebrity / athlete
  - Product / review
  - Tech company
  - Church
  - Homepage
  - ….

Amit Singhai says Google uses over 200 such features [NY Times 2008-06-03]

Generative vs. Discriminative

- 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

- 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

- 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

Ranking SVM

- Training data is

$$(q_1, r_1), (q_2, r_2), \ldots, (q_n, r_n)$$

  - r is partial rank information
    - if document d_a should be ranked higher than d_b, then $(d_a, d_b) \in r_i$
  - partial rank information comes from relevance judgments (allows multiple levels of relevance) or click data
    - e.g., d_1, d_2 and d_3 are the documents in the first, second and third rank of the search output, only d_3 clicked on → (d_3, d_1) and (d_3, d_2) will be in desired ranking for this query

Ranking SVM

- Learning a linear ranking function $\vec{w}.\vec{d}_a$
  - where *w* is a weight vector that is adjusted by learning
  - d_a is the vector representation of the features of document
  - *non-linear* functions also possible
- Weights represent importance of features
  - learned using training data
  - e.g.,

$$\vec{w}.\vec{d} = (2, 1, 2).(2, 4, 1) = 2.2 + 1.4 + 2.1 = 10$$

Ranking SVM

- Learn *w* that satisfies as many of the following conditions as possible:

$$\begin{aligned} \forall (d_i, d_j) \in r_1 &: \vec{w}.\vec{d_i} > \vec{w}.\vec{d_j} \\ &\ldots \\ \forall (d_i, d_j) \in r_n &: \vec{w}.\vec{d_i} > \vec{w}.\vec{d_j} \end{aligned}$$

- Can be formulated as an *optimization* problem

Ranking SVM

$$\begin{aligned}
&minimize: && \frac{1}{2}\vec{w}.\vec{w} + C\sum \xi_{i,j,k} \\
&subject\ to: && \\
&\forall(d_i, d_j) \in r_1 &:& \quad \vec{w}.\vec{d_i} > \vec{w}.\vec{d_j} + 1 - \xi_{i,j,1} \\
& && \ldots \\
&\forall(d_i, d_j) \in r_n &:& \quad \vec{w}.\vec{d_i} > \vec{w}.\vec{d_j} + 1 - \xi_{i,j,n} \\
& && \quad \forall i \forall j \forall k : \xi_i, j, k \geq 0
\end{aligned}$$

- *ξ*, known as a slack variable, allows for misclassification of difficult or noisy training examples, and *C* is a parameter that is used to prevent overfitting

Ranking SVM

- Software available to do optimization
- Each pair of documents in our training data can be represented by the vector:

$(\vec{d_i} - \vec{d_j})$

- Score for th $\vec{w}.(\vec{d_i} - \vec{d_j})$
- SVM classifi a *w* that makes the smallest score as large as possible
  - make the differences in scores as large as possible for the pairs of documents that are hardest to rank

Topic Models

- Improved representations of documents
  - can also be viewed as improved smoothing techniques
  - improve estimates for words that are related to the topic(s) of the document
    - instead of just using background probabilities
- Approaches
  - *Latent* Semantic Indexing (LSI)
  - Probabilistic *Latent* Semantic Indexing (pLSI)
  - *Latent* Dirichlet Allocation (LDA)

LDA

- Model document as being generated from a *mixture* of topics

1. For each document D, pick a multinomial distribution θ_D from a Dirichlet distribution with parameter α ,
2. For each word position in document D,
   (a) pick a topic z from the multinomial distribution θ_D ,
   (b) Choose a word w from $P(w|z, \beta)$, a multinomial probability conditioned on the topic z with parameter β.

LDA

- Gives language model probabilities

$$P_{lda}(w|D) = P(w|\theta_D, \beta) = \sum_z P(w|z, \beta)P(z|\theta_D)$$

- Used to smooth the document representation by mixing them with the query likelihood probability as follows:

$$P(w|D) = \lambda \left(\frac{f_{w,D} + \mu \frac{c_w}{|C|}}{|D| + \mu} \right) + (1 - \lambda)P_{lda}(w|D)$$

LDA

- If the LDA probabilities are used directly as the document representation, the effectiveness will be significantly reduced because the features are *too smoothed*
  - e.g., in typical TREC experiment, only 400 topics used for the *entire* collection
  - generating LDA topics is expensive
- When used for smoothing, effectiveness is improved

LDA Example

- Top words from 4 LDA topics from TREC news

Arts	*Budgets*	*Children*	*Education*
new	million	children	school
film	tax	women	students
show	program	people	schools
music	budget	child	education
movie	billion	years	teachers
play	federal	families	high
musical	year	work	public
best	spending	parents	teacher
actor	new	says	bennett
first	state	family	manigat
york	plan	welfare	namphy
opera	money	men	state
theater	programs	percent	president
actress	government	care	elementary
love	congress	life	haiti

Summary

- 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
- Query logs important for training ranker