Information Retrieval (IR)  
Result Ranking  

CSE 454

Class Overview

Other Cool Stuff
- Query processing
- Indexing
- IR - Ranking
- Content Analysis
- Crawling
- Document Level
- Learning Layer

A Closeup view

10/22 – IR - Ranking  
10/27 – Indexing  
10/29 – Alta Vista  
Pagerank  
11/3 – No class  
11/5 - Advertising

Standard Web Search Engine Architecture

Crawl the web  
Store documents, check for duplicates, extract links  
DocIds  
Create an inverted index  
Search engine servers show results To user

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

from Croft, Metzler, Strohman. © Addison Wesley
Test Corpora

<table>
<thead>
<tr>
<th>Collection</th>
<th>N</th>
<th>Docs</th>
<th>Size (MB)</th>
<th>TermDocs</th>
<th>Q-D Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADE</td>
<td>2309</td>
<td>4</td>
<td>2</td>
<td>400</td>
<td>&gt;10,000</td>
</tr>
<tr>
<td>CACM</td>
<td>3204</td>
<td>64</td>
<td>3</td>
<td>24.5</td>
<td></td>
</tr>
<tr>
<td>CISET</td>
<td>1460</td>
<td>112</td>
<td>3</td>
<td>46.5</td>
<td></td>
</tr>
<tr>
<td>CLEF</td>
<td>1,934</td>
<td>222</td>
<td>2</td>
<td>27.1</td>
<td></td>
</tr>
<tr>
<td>LISA</td>
<td>4072</td>
<td>32</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Madhur</td>
<td>1035</td>
<td>35</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSP</td>
<td>11,459</td>
<td>93</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CBIRMB</td>
<td>38,356</td>
<td>106</td>
<td>488</td>
<td>250</td>
<td>16.141</td>
</tr>
<tr>
<td>Reuters</td>
<td>21,578</td>
<td>672</td>
<td>28</td>
<td>134</td>
<td></td>
</tr>
<tr>
<td>TREC</td>
<td>260,000</td>
<td>200</td>
<td>2000</td>
<td>0.93429</td>
<td>&gt;10,000</td>
</tr>
</tbody>
</table>

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

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)

<table>
<thead>
<tr>
<th>Relevant</th>
<th>Not Relevant</th>
</tr>
</thead>
<tbody>
<tr>
<td>Retrieved</td>
<td>tp fp</td>
</tr>
<tr>
<td>Not Retrieved</td>
<td>fn tn</td>
</tr>
</tbody>
</table>

- Precision $P = \frac{tp}{tp + fp}$
- Recall $R = \frac{tp}{tp + fn}$

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

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
### 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.

### 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

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

### Term-document incidence

<table>
<thead>
<tr>
<th></th>
<th>Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Brutus</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mercy</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>worser</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

1 if play contains word, 0 otherwise
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) \( \Leftrightarrow \) bitwise AND.
  
  \[
  110100 \text{ AND } 110111 \text{ AND } 101111 = 100100.
  \]

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

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.

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**

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
Case folding

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

Thesauri and soundex

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

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?

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**

Stemming

- 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.

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](http://www.sims.berkeley.edu/~hearst/irbook/porter.html)
Typical rules in Porter

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

Challenges

- Sandy
- Sanded → Sand ???

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).

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.

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

Retrieval Model Overview

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

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

\[
|X \cap Y|
\]

<table>
<thead>
<tr>
<th>Antony and Cleopatra</th>
<th>Julius Caesar</th>
<th>The Tempest</th>
<th>Hamlet</th>
<th>Othello</th>
<th>Macbeth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Antony</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Brutus</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Caesar</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Calpurnia</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Cleopatra</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>mercy</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>worse</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**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

**Many Overlap Measures**

- Simple matching (coordination level match)
- Dice’s Coefficient
- Jaccard’s Coefficient
- Cosine Coefficient
- Overlap Coefficient

**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)

**Documents in 3D Space**

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

**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
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
- Vector distance measure used to rank retrieved documents

\[ D_k = w_{d_1}, w_{d_2}, ..., w_{d_t} \]
\[ Q = w_{q_1}, w_{q_2}, ..., w_{q_t} \]
\[ w = 0 \text{ if a term is absent} \]

Documents in 3D Space

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

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

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...
  - Need to consider "term frequency"—how many times the word appears in the current document.

TF x IDF

\[ w_{ik} = tf_{ik} \times \log \left( \frac{N}{n_k} \right) \]

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

Inverse Document Frequency

\[ \log \left( \frac{10000}{10000} \right) = 0 \]
\[ \log \left( \frac{10000}{10000} \right) = 0.301 \]
\[ \log \left( \frac{10000}{5000} \right) = 2.698 \]
\[ \log \left( \frac{10000}{20} \right) = 4 \]
TF-IDF normalization

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

$$w_{tk} = \frac{df_{tk} \log(N/n_t)}{\sqrt{\sum_{k=1}^{nNtf}(df_{tk})^2 \log(N/n_t)}}$$

Vector space similarity

(use the weights to compare the documents)

Now, the similarity of two documents is:

$$\text{sim}(D_i, D_j) = \sum_{k=1}^{nNtf} w_{tk} * w_{jk}$$

This is also called the cosine, or normalized inner product. (Normalization was done when weighting the terms.)

Computing Cosine Similarity Scores

- $D_1 = (0.8, 0.3)$
- $D_2 = (0.2, 0.7)$
- $Q = (0.4, 0.8)$
- $\cos \alpha_1 = 0.74$
- $\cos \alpha_2 = 0.98$

Computing a similarity score

Say we have query vector $Q = (0.4, 0.8)$
Also, document $D_3 = (0.2, 0.7)$
What does their similarity comparison yield?

$$\text{sim}(Q, D_3) = \frac{(0.4*0.2) + (0.8*0.7)}{\sqrt{(0.4)^2 + (0.8)^2} * (0.2)^2 + (0.7)^2}}$$

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

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?

Summary: Why use vector spaces?

- User’s query treated as a (very) short document.
- Query $\rightarrow$ 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.
**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."

**Bayes Classifier**

- Bayes Decision Rule
  - A document $D$ is relevant if $P(R|D) > P(NR|D)$
- Estimating probabilities
  - Use Bayes Rule
  - Classify $P(R|D) = \frac{P(D|R)P(R)}{P(D)}$ ant if
    - lhs is $\frac{P(D|R)}{P(D|NR)} > \frac{P(NR)}{P(R)}$

**Estimating $P(D|R)$**

- Assume independence
  - $P(D|R) = \prod_{i=1}^{n} P(d_i|R)$
- 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**

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

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

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

**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
  - $\log \frac{0.5(1-s_i)}{N(N-0.5)} = \log \frac{N-s_i}{N}$
Contingency Table

<table>
<thead>
<tr>
<th>Relevant</th>
<th>Non-relevant</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_i = 1$</td>
<td>$R_i$</td>
<td>$N - R_i$</td>
</tr>
<tr>
<td>$d_i = 0$</td>
<td>$R - R_i$</td>
<td>$N - R + R_i$</td>
</tr>
<tr>
<td>Total</td>
<td>$R$</td>
<td>$N - R$</td>
</tr>
</tbody>
</table>

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

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

Gives scoring function:

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

BM25

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

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

- $k_1$, $k_2$, and $K$ are parameters whose values are set empirically
- $K = k_1(1 - b) + k_2 \cdot \frac{dl}{avdl}$
- 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”, $(af = 1)$
- 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 ($dl/avdl = 0.9$)
- $k_1 = 1.2$, $b = 0.75$, and $k_2 = 100$
- $K = 1.2 \cdot (0.25 + 0.75 \cdot 0.9) = 1.11$

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
  $$p(D|Q) \propto P(Q|D)P(D)$$
- 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

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 $a_D$

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
Jelinek-Mercer Smoothing

- $\alpha_D$ is a constant, $\lambda$
- Gives estimate of
  \[ P(q_i | D) = (1 - \lambda) \frac{f_{q_i, D}}{|D|} + \lambda \frac{c_{q_i}}{|C|} \]
- Use $P(Q|D) = \prod_{i=1}^{n} (1 - \lambda) \frac{f_{q_i, D}}{|D|} + \lambda \frac{c_{q_i}}{|C|}$
- 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?

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

Dirichlet Smoothing

- $\alpha_D$ depends on document length
  \[ \alpha_D = \frac{\mu}{|D| + \mu} \]
- Gives probability estimation of
  \[ p(q_i | D) = \frac{f_{q_i, D} + \alpha_D}{|D| + \mu} \]
- And doc $P(q_i | D) = \frac{f_{q_i, D} + \alpha_D}{|D| + \mu}$

\[ \log P(Q|D) = \sum_{i=1}^{n} \log \left( \frac{f_{q_i, D} + \alpha_D}{|D| + \mu} \right) \]

Query Likelihood Example

For the term “president”
- $f_{q_i, D} = 15$, $c_{q_i} = 160,000$
- For the term “lincoln”
  - $f_{q_i, D} = 25$, $c_{q_i} = 2,400$
  - Number of word occurrences in the document $|D|$ is assumed to be 1,800
  - Number of word occurrences in the collection is $10^8$
  - 500,000 documents times an average of 2,000 words
  - $\mu = 2,000$

\[ QL(Q, D) = \log \frac{15 + 2,000 \times (1.6 \times 10^5) / (10^6)}{1,800 + 2,000} \]
\[ + \log \frac{25 + 2,000 \times (2,400) / (10^6)}{1,800 + 2,000} \]
\[ = \log(15.32/1800) + \log(25.005/1800) \]
\[ = 5.51 + 5.02 = -10.53 \]

- Negative number because summing logs of small numbers
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 KL-divergence, and assume relevance model \( R \) is the true distribution (not symmetric),
  \[
  \sum_{w \in V} P(w|R) \log P(w|D) - \sum_{w \in V} P(w|R) \log P(w|R) \]

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
  \[
  P(w, q_1 \ldots q_n) = \sum_{D \in \mathcal{D}} P(D) P(w, q_1 \ldots q_n|D)
  \]
- Assume
  \[
  P(w, q_1 \ldots q_n|D) = P(w|D) \prod_{i=1}^{n} P(q_i|D)
  \]
- Gives
  \[
  P(w, q_1 \ldots q_n) = \sum_{D \in \mathcal{D}} P(D) P(w|D) \prod_{i=1}^{n} P(q_i|D)
  \]
Estimating the Relevance Model

- \( P(D) \) usually assumed to be uniform
- \( P(w, q_1 \ldots q_n) \) 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
- 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 \( C \).
3. Calculate the relevance model probabilities \( P(w|Q) \). \( P(Q_1 \ldots Q_n) \) is used as a normalizing constant and is calculated as
   \[
   P(Q_1 \ldots Q_n) = \sum_{w \in \Omega} P(w|Q_1 \ldots Q_n)
   \]
4. Rank documents again using the KL-divergence score:
   \[
   \sum_{w \in \Omega} P(w|Q) \log P(w|D)
   \]

Example from Top 10 Docs

<table>
<thead>
<tr>
<th>president</th>
<th>lincoln</th>
<th>washington</th>
<th>bush</th>
<th>reagan</th>
<th>clinton</th>
</tr>
</thead>
<tbody>
<tr>
<td>friend</td>
<td>fish</td>
<td>seal</td>
<td>seal</td>
<td>seal</td>
<td>seal</td>
</tr>
</tbody>
</table>

Example from Top 50 Docs

<table>
<thead>
<tr>
<th>president</th>
<th>lincoln</th>
<th>washington</th>
<th>bush</th>
<th>reagan</th>
<th>clinton</th>
</tr>
</thead>
<tbody>
<tr>
<td>friend</td>
<td>fish</td>
<td>seal</td>
<td>seal</td>
<td>seal</td>
<td>seal</td>
</tr>
</tbody>
</table>

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 $\theta$ 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)

- **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 $P(I|D, \mu)$

---

**Example: AND Combination**

$D$ and $E$ are parent nodes for $Q$

<table>
<thead>
<tr>
<th>$P(Q) = \text{TRUE}_{[a, b]}$</th>
<th>$a$</th>
<th>$b$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>0</td>
<td>FALSE</td>
<td>TRUE</td>
</tr>
<tr>
<td>1</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>1</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

**Inference Network Operators**

- $bel_{\text{null}}(q) = 1 - p_{q}$
- $bel_{\text{not}}(q) = 1 - \prod (1 - p_{r})$
- $bel_{\text{and}}(q) = \prod p_{r}$
- $bel_{\text{or}}(q) = \prod p_{r}$
- $bel_{\text{max}}(q) = \max \{p_{r_1}, p_{r_2}, \ldots, p_{r_n}\}$
- $bel_{\text{min}}(q) = \prod r_{r_i}$
- $bel_{\text{mean}}(q) = \sum p_{r_i}$
- $bel_{\text{median}}(q) = \frac{\sum \text{weight}(p_r)}{\sum \text{weight}(r)}$

---

**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{adult stem cells} @combine{embryonic stem cells}
  0.1 @combine{adult stem cells} @combine{embryonic stem cells}
  #uw8(embryonic stem) #uw12(embryonic stem cells)
  ```

Example Web Query

```weight{
0.8 @weight{0.6 @prior pagerank} 0.4 @prior(stlinks)}
1.0 @weight{
  0.9 @combine{
    @weight{1.0 @prior pagerank} 1.0 @prior(stlinks)}
  0.9 @combine{
    @weight{3.0 @prior pagerank} 1.0 @prior(stlinks)}
  0.9 @combine{
    @weight{3.0 @prior pagerank} 1.0 @prior(stlinks) 1.0 @prior pagerank}"
```

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.

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

Learning a linear ranking function \( \mathbf{w} \cdot \mathbf{d} \),
where \( \mathbf{w} \) is a weight vector that is adjusted by learning.
\( \mathbf{d} \) 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.,
\[
\mathbf{w} \cdot \mathbf{d} = (2, 1, 2) \cdot (2, 4, 1) = 2 \times 2 + 1 \times 4 + 2 \times 1 = 10
\]
Ranking SVM

\[
\begin{align*}
\text{minimize:} & \quad \frac{1}{2} \mathbf{w}^T \mathbf{w} + C \sum \xi_{i,j,k} \\
\text{subject to:} & \quad \forall (d_i, d_j) \in r_1 : \quad \langle w, d_i \rangle - \langle w, d_j \rangle + 1 - \xi_{i,j,1} \\
& \quad \ldots \\
& \quad \forall (d_i, d_j) \in r_n : \quad \langle w, d_i \rangle - \langle w, d_j \rangle + 1 - \xi_{i,j,n} \\
& \quad \forall i, j, k \geq 0
\end{align*}
\]

- \( \xi \), 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.

**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 \( \theta_D \) from a Dirichlet distribution with parameter \( \alpha \)
  2. For each word position in document \( D_i \)
    - (a) pick a topic \( z \) from the multinomial distribution \( \theta_D \)
    - (b) Choose a word \( w \) from \( P(w|z, \beta) \), a multinomial probability conditioned on the topic \( z \) with parameter \( \beta \)

- 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:
  \[ P(w|D) = \lambda \left( \frac{F(w,D,w|\theta_D)}{|D|+\mu} \right) + (1-\lambda)P_{lda}(w|D) \]

- 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

<table>
<thead>
<tr>
<th>Arts</th>
<th>Budget</th>
<th>Children</th>
<th>Education</th>
</tr>
</thead>
<tbody>
<tr>
<td>arts</td>
<td>billion</td>
<td>children</td>
<td>school</td>
</tr>
<tr>
<td>music</td>
<td>tax</td>
<td>resources</td>
<td>scientific</td>
</tr>
<tr>
<td>show</td>
<td>program</td>
<td>people</td>
<td>schools</td>
</tr>
<tr>
<td>movie</td>
<td>budget</td>
<td>child</td>
<td>education</td>
</tr>
<tr>
<td>play</td>
<td>billion</td>
<td>years</td>
<td>teachers</td>
</tr>
<tr>
<td>musical</td>
<td>year</td>
<td>work</td>
<td>public</td>
</tr>
<tr>
<td>house</td>
<td>spending</td>
<td>parents</td>
<td>teacher</td>
</tr>
<tr>
<td>actor</td>
<td>new</td>
<td>says</td>
<td>honest</td>
</tr>
<tr>
<td>first</td>
<td>style</td>
<td>family</td>
<td>manipulate</td>
</tr>
<tr>
<td>work</td>
<td>plan</td>
<td>welfare</td>
<td>empty</td>
</tr>
<tr>
<td>opera</td>
<td>money</td>
<td>new</td>
<td>state</td>
</tr>
<tr>
<td>theatre</td>
<td>propose</td>
<td>percent</td>
<td>president</td>
</tr>
<tr>
<td>actress</td>
<td>government</td>
<td>core</td>
<td>elementary</td>
</tr>
<tr>
<td>love</td>
<td>congress</td>
<td>file</td>
<td>halt</td>
</tr>
</tbody>
</table>

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