Chapter 7

Retrieval Models
Retrieval Models

- Provide a mathematical framework for defining the search process
  - Includes explanation of assumptions
  - Basis of many ranking algorithms
- Progress in retrieval models has corresponded with improvements in effectiveness
- Theories about 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
Retrieval Model Overview

- Older models
  - Boolean retrieval
  - Vector Space model

- Probabilistic Models
  - BM25
  - Language models

- Combining evidence
  - Inference networks
  - Learning to Rank
Searching Based on Retrieved Documents

- Sequence of queries driven by the number of retrieved documents

Example. “Lincoln” search of news articles

- president AND Lincoln
- president AND Lincoln AND NOT (automobile OR car)
- president AND Lincoln AND biography AND life AND birthplace AND gettysburg AND NOT (automobile OR car)
- president AND Lincoln AND (biography OR life OR birthplace OR gettysburg) AND NOT (automobile OR car)
Boolean Retrieval

- Two possible outcomes for query processing
  - TRUE and FALSE
  - “Exact-match” retrieval
  - No ranking at all

- Query usually specified using Boolean operators
  - AND, OR, NOT
Boolean Retrieval

- **Advantages**
  - Results are *predictable*, relatively easy to explain
  - Many different *document features*, such as metadata (e.g., type/date), can be incorporated
  - *Efficient processing*, since many documents can be eliminated from search

- **Disadvantages**
  - Simple queries usually *don’t* work well
  - *Complex* queries are *difficult* to construct
  - *Effectiveness* depends entirely on users
Vector Space Model

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

\[ D_i = (d_{i1}, d_{i2}, \ldots, d_{it}) \quad \quad Q = (q_1, q_2, \ldots, q_t) \]

<table>
<thead>
<tr>
<th>Term_1</th>
<th>Term_2</th>
<th>\ldots</th>
<th>Term_t</th>
</tr>
</thead>
<tbody>
<tr>
<td>Doc_1</td>
<td>\text{d}_{11}</td>
<td>\text{d}_{12}</td>
<td>\ldots</td>
</tr>
<tr>
<td>Doc_2</td>
<td>\text{d}_{21}</td>
<td>\text{d}_{22}</td>
<td>\ldots</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\ldots</td>
</tr>
<tr>
<td>Doc_n</td>
<td>\text{d}_{n1}</td>
<td>\text{d}_{n2}</td>
<td>\ldots</td>
</tr>
</tbody>
</table>
## 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

<table>
<thead>
<tr>
<th>Terms</th>
<th>( D_1 )</th>
<th>( D_2 )</th>
<th>( D_3 )</th>
<th>( D_4 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>aquarium</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>bowl</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>care</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>fish</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>freshwater</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>goldfish</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>homepage</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>keep</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>setup</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>tank</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>tropical</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>
Vector Space Model

- 3-D pictures useful, but can be misleading for high-dimensional space
Vector Space Model

- Documents ranked by *differences* between points representing *query* and *documents*
  - Using *similarity measure* rather than a distance or *dissimilarity* measure
  - e.g. Cosine correlation

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

Consider two documents $D_1$ and $D_2$, and a query $Q$

- $D_1 = (0.5, 0.8, 0.3)$
- $D_2 = (0.9, 0.4, 0.2)$
- $Q = (1.5, 1.0, 0)$

$Cosine(D_1, Q) = \frac{(0.5 \times 1.5) + (0.8 \times 1.0)}{\sqrt{(0.5^2 + 0.8^2 + 0.3^2)(1.5^2 + 1.0^2)}}$

$= \frac{1.55}{\sqrt{(0.98 \times 3.25)}} = 0.87$

$Cosine(D_2, Q) = \frac{(0.9 \times 1.5) + (0.4 \times 1.0)}{\sqrt{(0.9^2 + 0.4^2 + 0.2^2)(1.5^2 + 1.0^2)}}$

$= \frac{1.75}{\sqrt{(1.01 \times 3.25)}} = 0.97$

More similar to $Q$ than $D_1$
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:
    \[
    idf_k = \log \frac{N}{n_k}
    \]
  - Some heuristic modifications
    \[
    d_{ik} = \frac{(\log(tf_{ik}) + 1) \cdot \log(N/n_k)}{\sqrt{\sum_{k=1}^{t} [(\log(tf_{ik}) + 1.0) \cdot \log(N/n_k)]^2}}
    \]
  - Ensure non-zero weight
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 *assumption* on whether relevance is binary/multivalued
Probabilistic Models

- According to [Greiff 02]
  - In probabilistic approaches to IR, the occurrence of a query term in a document $D$ contributes to the probability that $D$ will be judged as relevant.
  - The weight assigned to a query term should be based on the expected value of that contribution.

IR as Classification

Bayes Decision Rule
Based on Bayes Classifier
Bayes Classifier

- **Bayes Decision Rule**
  
  A document $D$ is **relevant** if $P(R \mid D) > P(NR \mid D)$, where $P(R \mid D)$ and $P(NR \mid D)$ are conditional probabilities.

- **Estimating probabilities**
  
  - Use Bayes Rule
    
    $$P(R \mid D) = \frac{P(D \mid R)P(R)}{P(D)}$$
    
    - $P(D \mid R)P(R)$: Likelihood ratio of $D$ being relevant based on the probability of occurrence of words in $D$ that are in $R$.
    - $P(D)$: A prior probability of relevance.
  
  - Classify a document as **relevant** if
    
    $$\frac{P(D \mid R)}{P(D \mid NR)} > \frac{P(NR)}{P(R)}$$
    
    Based on the Bayes Decision Rule $P(R \mid D) > P(NR \mid D)$.
    
    - L.H.S. is the **likelihood ratio** of $D$ being relevant.
Estimating $P(D \mid R)$

- Assume word independence (Naïve Bayes assumption) and use individual term/word ($d_i$) probabilities

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

- Binary (weights in doc) independence (of word) model
  - Document represented by a vector of binary features indicating term occurrence (or non-occurrence)
  - $p_i$ is probability that term $i$ occurs in relevant document, $s_i$ is probability of occurrence of $i$ in non-relevant document
  - **Example.** Given a document representation $(1, 0, 0, 1, 1)$, the probability of the document occurring in the relevant set is $$p_1 \times (1 - p_2) \times (1 - p_3) \times p_4 \times p_5$$
Binary Independence Model

- Computing the likelihood ratio of $D$ using $p_i$ and $s_i$

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

Probability of $i$ in $R$  
Probability of $i$ in $NR$  
Probability of $i$ not in $R$  
Probability of $i$ not in $NR$

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

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

(Same for all documents)
Binary Independence Model

- Scoring function is

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

- Using \( \log \) to avoid multiplying lots of small numbers

- A query \( Q \) provides information about relevant docs

- If there is no other information on the relevant set
  - \( p_i \) is set to be a constant (= 0.5)
  - \( s_i \) is approximated as \( n_i / N \), where \( n_i \) is the number of documents in the collection \( N \) that include \( i \)
  - \( s_i \) is similar to the \textit{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, If (non-)Relevant Sets available

<table>
<thead>
<tr>
<th></th>
<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_i - r_i )</td>
<td>( n_i )</td>
</tr>
<tr>
<td>( d_i = 0 )</td>
<td>( R - r_i )</td>
<td>( N - n_i - R + r_i )</td>
<td>( N - n_i )</td>
</tr>
<tr>
<td>Total</td>
<td>( R )</td>
<td>( N - R )</td>
<td>( N )</td>
</tr>
</tbody>
</table>

where \( r_i = \) number of relevant documents containing term \( i \)
\( n_i = \) number of documents in a collection \( N \) containing \( i \)
\( R = \) number of relevant documents in the collection
\( N = \) number of documents in the collection

\[
p_i = \frac{r_i + 0.5}{R + 1} \quad s_i = \frac{n_i - r_i + 0.5}{N - R + 1}
\]

Gives the **scoring** function:

\[
\sum_{i: d_i = 1} \log \frac{p_i (1 - s_i)}{s_i (1 - p_i)} \\
\approx \sum_{i: d_i = q_i = 1} \log \frac{(r_i + 0.5)/(R - r_i + 0.5)}{(n_i - r_i + 0.5)/(N - n_i - R + r_i + 0.5)}
\]
BM25 Ranking Algorithm

- A *ranking* algorithm based on *probabilistic arguments* and *experimental validation*, but it is not a formal model

- Adds document and query term *weights*

\[
\sum_{i \in Q} \log \frac{(r_i + 0.5)}{(n_i - r_i + 0.5)} \cdot \frac{(k_1 + 1)f_i}{K + f_i} \cdot \frac{(k_2 + 1)qf_i}{k_2 + qf_i}
\]

- \(f_i\) is the *frequency of term* *i* in a document *D*

- \(k_1\), a constant, plays the role of \(tf\) & changes as \(f_i\) increases

- \(qf_i\) is the frequency of term *i* in query *Q*, much lower than \(f_i\)

- \(k_2\) plays the role as \(k_1\) but is applied to query term weight \(qf_i\)

- \(K = k_1((1 - b) + b \cdot \frac{dl}{avdl})\), where \(dl\) is document length & \(avdl\) is a *normalization factor* of \(tf\) regulated by \(b\) (0 ≤ \(b\) ≤ 1)

- \(k_1, k_2\) & \(K\) are set empirically as 1.2, 0..1000, & 0.75, resp.
BM25 Example

- Given a query with two terms, “president lincoln”, \((qf = 1)\)
- No relevance information \((r_i \text{ and } R \text{ are zero})\)
- \(N = 500,000\) 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 \(D\) \((f_1 = 15)\)
- “lincoln” occurs 25 times in doc \(D\) \((f_2 = 25)\)
- Document length is 90% of the average length \((dl / avdl = 0.9)\)
- \(k_1 = 1.2, \ b = 0.75, \text{ and } k_2 = 100\)
- \(K = 1.2 \times (0.25 + 0.75 \times 0.9) = 1.11\)
BM25 Example

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

- Effect of term frequencies, especially on “lincoln”

<table>
<thead>
<tr>
<th>Frequency of “president”</th>
<th>Frequency of “lincoln”</th>
<th>BM25 score</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>25</td>
<td>20.66</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>12.74</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>5.00</td>
</tr>
<tr>
<td>1</td>
<td>25</td>
<td>18.2</td>
</tr>
<tr>
<td>0</td>
<td>25</td>
<td>15.66</td>
</tr>
</tbody>
</table>
A (statistical) **language model** assigns a probability to a sequence of *words* by means of a probability distribution, i.e., puts a probability measure over strings from some vocabulary, \( \sum_{s \in \Sigma^*} P(s) = 1 \)

**LM**, which is based on the statistical theory and NLP, have been successfully applied to the problem of ad-hoc IR retrieval.

The original and basic method for using LMs in IR is the **query likelihood model** in which a document \( d \) is constructed from a **language model** \( M_d \).

LM approaches *estimate* a LM for each *document* \( d \) & then *rank* documents by the likelihood of the *query* according to the estimated LM, i.e., \( P(Q \mid M_d) \).
Three possibilities models of topical relevance

- Probability of generating the query text from a document (language) model, i.e., the query likelihood model:

\[ P_{mle}(Q \mid M_d) = \prod_{t \in Q} P_{mle}(t \mid M_d), \text{ where } P_{mle}(t \mid M_d) = \frac{tf_{t,d}}{dl_d} \]

- Probability of generating the document text from a query (language) model, i.e.,

\[ P_{mle}(D \mid M_q) = P_{mle}(D \mid R), \text{ where } R \text{ is the set of relevant documents of a query } q \]

- Comparing the language models representing the query and document topics
The Query Likelihood Model

- A separate *language model* is associated with each doc in a collection
- Documents are ranked based on the *probability* of the query $Q$ in the document (language) model, $P(Q \mid M_d)$
- Assumption: a query $Q$ is “generated” by a probability model based on a document $D$

  - Given a query $Q = q_1 \ldots q_n$ and a document $D = d_1 \ldots d_m$, compute the *conditional probability* $P(D \mid Q)$

$$P(D \mid Q) = P(Q \mid D) \cdot P(D)$$

where $P(Q \mid D)$ $P(D)$ is the *query likelihood* given $D$ & $P(D)$ is the prior belief that $D$ is *relevant* to any query
Query-Likelihood Model

- Rank documents, \( P(D \mid Q) \), by the probability that the query could be generated by the document model (i.e., on the same topic), \( P(Q \mid D) \)
- Given a query \( Q \), start with \( P(D \mid Q) \)
- Using Bayes’ Rule
- Assuming prior is uniform, unigram model

\[
p(D \mid Q)_{\text{rank}} = P(Q \mid D) P(D)
\]

(Assumed to be uniform)

\[
P(Q \mid D) = \prod_{i=1}^{n} P(q_i \mid D)
\]
Estimating Probabilities

- Obvious estimate for *unigram probabilities* is

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

- **Maximum likelihood estimate (mle)**
  - Makes the observed value of \( f_{q_i, D} \) most likely
  - If any query words are missing from document, score will be zero
    - Missing 1 out of 4 query words same as missing 3 out of 4
LMs for Retrieval

The Document Likelihood Model

- Instead of using the probability of a *document language model* \( (M_d) \) generating the *query*, use the probability of a *query language model* \( M_q \) generating the *document*

- Creating a *document likelihood model*, even though it is less appealing, since there is much less text available to estimate a LM based on the query text

- Solution: incorporate relevance feedback into the model & update the language model \( M_q \)

  - e.g., the *relevance model* of Lavrenko & Croft (2001) is an instance of a document likelihood model that incorporates *pseudo relevance feedback* into an LM approach
Comparing the Query Likelihood Model & Document Likelihood Model

Rather than generating either a document language model ($M_d$) or a query language model ($M_q$), create an LM from both the document & query & find the difference.

To model the risk of returning a document $d$ as relevant to a query $q$, use the KL-divergence

$$R(d; q) = \text{KL}(M_d \parallel M_q) = \sum_{t \in V} P(t \mid M_q) \log \frac{P(t \mid M_q)}{P(t \mid M_d)}$$

- KL-divergence measures how bad the probability distribution $M_q$ is at modeling $M_d$
- It has been shown that the comparison model outperforms both query-likelihood & document-likelihood models and useful for ad hoc retrieval.
LMs for Retrieval

Three ways of developing the LM modeling approach:

a) The Query Likelihood Model: $P(Q \mid M_q)$ or $P(Q \mid D)$

b) The Document Likelihood Model: $P(D \mid M_Q)$ or $P(D \mid R)$

c) Comparison of the query and document LMs using $KL$-Divergence
Smoothing

- Document texts are a *sample* from the *language model*
  - Missing words (in a document) should *not* have *zero* probability of occurring

- **Smoothing** is a technique for estimating probabilities for missing (or unseen) words for the words that are *not seen* in the text
  - Lower (or *discount*) the *probability* estimates for words that are *seen* in the document text
  - Assign that “left-over” probability to the estimates
As stated in [Zhai 04]

- **Smoothing** is the problem of *adjusting* the *maximum likelihood estimator* to compensate for *data sparseness* and it is required to avoid assigning *zero probability* to *unseen* words.

- Smoothing accuracy is directly related to the retrieval performance:
  - The retrieval performance is generally *sensitive* to *smoothing parameters*.

- Smoothing plays two different roles:
  - Making the estimated document LM more *accurate*.
  - Explaining the *non-informative* words in the query.
Estimating Probabilities

- Estimate for unseen words is $\alpha_D P(q_i \mid C)$
  
  - $P(q_i \mid C)$ is the probability for query word $i$ in the collection language model for collection $C$ (background probability)
  
  - $\alpha_D$, which can be a constant, is a coefficient of the probability assigned to unseen words, depending on the document $D$

- Estimate for words that occur is
  
  $$(1 - \alpha_D) P(q_i \mid D) + \alpha_D P(q_i \mid C)$$

- Different forms of estimation come from different $\alpha_D$

<table>
<thead>
<tr>
<th>Method</th>
<th>$p_s(w \mid d)$</th>
<th>$\alpha_d$</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jelinek-Mercer</td>
<td>$(1 - \lambda) p_{ml}(w \mid d) + \lambda p(w \mid C)$</td>
<td>$\lambda$</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>Dirichlet</td>
<td>$\frac{c(w; d) + \mu p(w \mid C)}{\sum_w c(w; d) + \mu}$</td>
<td>$\mu$</td>
<td>$\mu$</td>
</tr>
</tbody>
</table>
Estimating Probabilities

Different forms of estimation come from different $\alpha_D$ (or $\lambda$)

<table>
<thead>
<tr>
<th>Method</th>
<th>$p_s(w \mid d)$</th>
<th>$\alpha_d$</th>
<th>Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jelinek-Mercer</td>
<td>$(1 - \lambda) p_{ml}(w \mid d) + \lambda p(w \mid C)$</td>
<td>$\lambda$</td>
<td>$\lambda$</td>
</tr>
<tr>
<td>Dirichlet</td>
<td>$\frac{c(w; d) + \mu p(w \mid C)}{\sum_w c(w; d) + \mu}$</td>
<td>$\mu$</td>
<td>$\mu$</td>
</tr>
</tbody>
</table>

where $p_s(w \mid d)$ is the smoothed probability of a word seen in $d$, $p(w \mid C)$ is the collection language model, $\alpha_d$ is a coefficient controlling the probability mass assigned to unseen words, all probabilities sum to one.

$$\alpha_d = \frac{1 - \sum_{w:c(w;d)>0} p_s(w \mid d)}{1 - \sum_{w:c(w;d)>0} p(w \mid C)}$$
Jelinek-Mercer Smoothing

- $\alpha_D$ is a constant, $\lambda$ ($= 0.1$ for short queries or $= 0.7$ for long queries in TREC evaluations)

- Gives estimate of

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

- Ranking score

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

- Use logs for convenience: solve the accuracy problem of multiplying small numbers

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

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

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

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

\[
\text{rank} \equiv \sum_{i:f_{q_i,D} > 0} \log \left( \frac{(1 - \lambda) \frac{f_{q_i,D}}{|D|} + \lambda \frac{c_{q_i}}{|C|}}{\lambda \frac{c_{q_i}}{|C|}} + 1 \right)
\]

- Proportional to the \textit{term frequency (TF)}, inversely proportional to the \textit{collection frequency (IDF)}
Dirichlet Smoothing

- $\alpha_D$ depends on document length

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

where $\mu$ is a parameter value set empirically

- 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.** Given \( \log P(Q|D) = \sum_{i=1}^{n} \log \frac{f_{q_i,D} + \mu c_{q_i}}{|D| + \mu} \),

- Let \( Q \) be “President Lincoln”, \( D \) (a doc) in \( C \)ollection
- For the term “President”, let \( f_{q_i,D} = 15 \), \( c_{q_i} = 160,000 \)
- For the term “Lincoln”, let \( f_{q_i,D} = 25 \), \( c_{q_i} = 2,400 \)
- Let no. of word occurrences in \( D \) (i.e., \( |D| \)) be 1,800
- Let no. of word occurrences in \( C \) be \( 10^9 = 500,000 \) (docs) \( \times 2,000 \) (average no. of words in a doc)
- \( \mu = 2,000 \)

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

Negative due to the summatting logs of small numbers
Example (Continued).

<table>
<thead>
<tr>
<th>Frequency of “president”</th>
<th>Frequency of “lincoln”</th>
<th>QL score</th>
</tr>
</thead>
<tbody>
<tr>
<td>15</td>
<td>25</td>
<td>-10.53</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>-13.75</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
<td>-19.05</td>
</tr>
<tr>
<td>1</td>
<td>25</td>
<td>-12.99</td>
</tr>
<tr>
<td>0</td>
<td>25</td>
<td>-14.40</td>
</tr>
</tbody>
</table>
Relevance Models

- **Relevance model**
  - Estimates a relevance model as the *query language model* and top-ranked documents
  - Uses queries (small) & relevant documents (large) as text samples generated from the relevance model
  - Predicts the relevance of new documents using some samples, i.e., relevance documents, for a query, \( P(D | R) \)

- \( P(D | R) \): *probability* of generating the text in a document \( D \) given a relevance model \( R \)
  - The *document likelihood model*
  - *Less effective* than the *query likelihood model* due to the difficulties of comparing documents of various sizes
Pseudo-Relevance Feedback

- Documents can be ranked by the similarity of the document model, $P(Q \mid M_d)$, to the relevance model, $P(D \mid R)$, originated from $P(D \mid M_q)$

- Question: “How to compare the two models, document model vs. relevance model?”

- Kullback-Leibler divergence (KL-divergence) is a well-known measure of the difference between two probability distributions

  \[
  KL(P \parallel Q) = \sum_{t \in V} P(t \mid P) \log \frac{P(t \mid P)}{P(t \mid Q)}
  \]
KL (Information) - Divergence

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

\[
KL(P \| Q) = \sum_{t \in V} P(t \mid P) \log \frac{P(t \mid P)}{P(t \mid Q)}
\]

- is a measure of the information *lost* when $Q$ is used to approximate $P$

- measures how *bad* the probability distribution $Q$ is at modeling $P$
KL (Information) - Divergence

- Given the *true* probability distribution \( P \) and distribution \( Q \) that is an *approximation* to \( P \)

\[
KL(P \mid\mid Q) = \sum_{t \in V} P(t \mid P) \log \frac{P(t \mid P)}{P(t \mid Q)}
\]

- Let \( R \) be the *relevance model* for a *query* and let \( D \) be the *document model* (in the *query likelihood model*). The negative KL-divergence for ranking is expressed as

\[
\sum_{w \in V} P(w \mid R) \log P(w \mid D) - \sum_{w \in V} P(w \mid R) \log P(w \mid R)
\]

Not depended on \( D \): ignore for ranking

where \( R \) is the *true* distribution & \( D \) is its *approximation*

- *Smaller* differences mean *higher* scores

- KL-divergence is *not* symmetric, i.e., \( KL(P \mid\mid Q) \neq KL(Q \mid\mid P) \)
KL-Divergence

- As stated earlier, \( \sum_{w \in V} P(w | R) \log P(w | D) \)

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

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

- rank-equivalent to *query likelihood score*, since \( V = Q \), when \( w \in Q \) & \( w \not\in Q \Rightarrow f_{w,Q} = 0 \)

- *Query likelihood model* is a special case of retrieval based on *relevance model*
Estimating the Relevance Model

- Alternatively, it is not restricted to the simple method of estimating the relevance model, i.e., \( P(w \mid R) \), using query term frequencies.

- 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.

\[
P(w \mid R) \approx P(w \mid q_1 \ldots q_n)
\]

- By definition, \( P(w \mid R) \) can be expressed as the joint probability of observing \( w \) with \( q_1, \ldots, q_n \), i.e.,

\[
P(w \mid R) \approx \frac{P(w, q_1 \ldots q_n)}{P(q_1 \ldots q_n)} = \sum_{w \in V} P(w, q_1 \ldots q_n)
\]
Estimating the Relevance Model

- The joint probability of $P(w, q_1 \ldots q_n)$ is
  \[
P(w, q_1 \ldots q_n) = \sum_{D \in C} p(D) P(w, q_1 \ldots q_n | D)
  \]
  where $C$ is a set of documents represented by LM

- Assume that
  \[
P(w, q_1 \ldots q_n | D) = P(w | D) \prod_{i=1}^{n} P(q_i | D)
  \]

- Thus, the joint probability of $P(w, q_1 \ldots q_n)$ is
  \[
P(w, q_1 \ldots q_n) = \sum_{D \in C} P(D) P(w | D) \prod_{i=1}^{n} P(q_i | D)
  \]
Estimating the Relevance Model

- Given that
  \[ P(w, q_1 \ldots q_n) = \sum_{D \in C} P(D) P(w|D) \prod_{i=1}^{n} P(q_i|D) \]
  - \( P(D) \) usually assumed to be uniform & thus is ignored
  - \( \prod_{i=1}^{n} P(q_i | D) \) is the query likelihood (QL) score for \( D \)
Set Theoretic Models

- The Boolean model imposes a *binary* criterion for deciding relevance
- The question of how to extend the Boolean model to accommodate *partial matching* and a *ranking* has attracted considerable attention in the past
- Two set theoretic models for this extension are
  - Fuzzy Set Model
  - Extended Boolean Model
Fuzzy Queries

- **Binary queries** are based on *binary logic*, in which a document either *satisfies* or does *not satisfy* a query.

- **Fuzzy queries** are based on *fuzzy logic*, in which a query term is considered as a fuzzy set & each document is given a degree of relevance with respect to a query, usually [0 – 1].

- **Example.**
  - In *binary logic*: The set of "tall people" are defined as those > 6 feet. Thus, a person 5' 9.995" is *not* considered tall.
  - In *fuzzy logic*: no clear distinction on “tall people.” Every person is given a *degree of membership* to the set of tall people, e.g.,
    - a person 7'0" will have a grade 1.0
    - a person 4'5" will have a grade 0.0
    - a person 6‘2" will have a grade 0.85
    - a person 5’8" will have a grade 0.7
Fuzzy Queries

- Binary Logic vs. Fuzzy Logic

Binary (crisp) Logic

Fuzzy Logic
Fuzzy Set Model

- Queries and documents are represented by sets of index terms & matching is approximate from the start.

- This vagueness can be modeled using a fuzzy framework, as follows:
  - with each (query) term is associated a fuzzy set
  - each document has a degree of membership \(0 \leq X \leq 1\) in this fuzzy set

- This interpretation provides the foundation for many models for IR based on fuzzy theory.

- The model proposed by Ogawa, Morita & Kobayashi (1991) is discussed.
Fuzzy Information Retrieval

- *Fuzzy sets* are modeled based on a *thesaurus*
- This thesaurus is built as follows:
  - Let $c$ be a *term-term correlation matrix* (or *keyword connection matrix*)
  - Let $c(i, l)$ be a *normalized correlation factor* for $(k_i, k_l)$:
    \[
    c(i, l) = \frac{n(i, l)}{n_i + n_l - n(i, l)}
    \]
    - $n_i$: number of documents which contain $k_i$
    - $n_l$: number of documents which contain $k_l$
    - $n(i, l)$: number of documents which contain both $k_i$ and $k_l$
- The notion of *proximity* among indexed terms
The correlation factor $c(i, l)$ can be used to define fuzzy set membership for a document $d_j$ in the fuzzy set for query term $K_i$ as follows:

$$
\mu(i, j) = 1 - \prod_{k_i \in d_j} (1 - c(i, l))
$$

- $\mu(i, j)$: the degree of membership of document $d_j$ in fuzzy subset associated with $k_i$

The above expression computes an algebraic sum over all terms in $d_j$ with respect to $k_i$, and is implemented as the complement of a negated algebraic product.

A document $d_j$ belongs to the fuzzy set for $k_i$, if its own terms are associated with $k_i$. 
Fuzzy Information Retrieval

\[ \mu(i, j) = 1 - \prod_{k_i \in d_j} (1 - c(i, l)) \]

- \( \mu(i, j) \): membership of document \( d_j \) in fuzzy subset associated with query term \( k_i \)

- If document \( d_j \) contains any index term \( k_i \) which is closely related to query term \( k_i \), we have
  - if \( \exists l \) such that \( c(i, l) \approx 1 \), then \( \mu(i, j) \approx 1 \), and
  - query index \( k_i \) is a good fuzzy index for document \( d_j \)
Example.

- $q = k_a \land (k_b \lor \neg k_c)$
- $q_{dnf} = (1,1,1) + (1,1,0) + (1,0,0)$ (binary weighted)
  - $= cc_1 + cc_2 + cc_3$ (conjunctive comp)
- $\mu(q, d_j) = \mu(cc_1 + cc_2 + cc_3, j)$
  - $= 1 - (1 - \mu(a, j) \mu(b, j) \mu(c, j)) \times$
    - $(1 - \mu(a, j) \mu(b, j) (1 - \mu(c, j))) \times$
    - $(1 - \mu(a, j) (1 - \mu(b, j)) (1 - \mu(c, j)))$
Fuzzy IR

Example.

- \( q = k_a \land (k_b \lor \neg k_c) \)
- \( q_{dnf} = (1,1,1) + (1,1,0) + (1,0,0) \) (binary weighted)
  \( = cc_1 + cc_2 + cc_3 \) (conjunctive comp)
- \( \mu(q, d_j) = \mu(cc_1 + cc_2 + cc_3, j) \)
  \( = 1 - (1 - \mu(a, j) \mu(b, j) \mu(c, j)) \times (1 - \mu(a, j) \mu(b, j) (1 - \mu(c, j))) \times (1 - \mu(a, j) (1 - \mu(b, j)) (1 - \mu(c, j))) \)

\( \mu \), the degree of membership, in a disjunctive fuzzy set is computed by using an algebraic sum, which is implemented as complement of a negated algebraic product, instead of the more common max function.
Fuzzy Information Retrieval

- *Fuzzy set* are useful for representing *vagueness* and *imprecision*

- *Fuzzy IR* models have been discussed mainly in the literature associated with *fuzzy theory*

- Experiments with standard test collections are not available

- Difficult to *compare* because previously conducted experiences used only small collections