Advanced IR models

Probabilistic model
Latent semantic indexing
Probabilistic Model

- An initial set of documents is retrieved (somehow)
- User inspects these docs looking for the relevant ones (only top 10-20) (we see later that we eliminate this manual step in the actual probabilistic model)
- IR system uses this info to refine description of ideal answer set
- By repeating this process, description of the ideal answer set will improve
- Description of ideal answer set is modeled in probabilistic terms
Probabilistic Ranking Principle

• Given a user query $q$ and a document $d_j$, the probabilistic model estimates the probability that the user will find the document $d_j$ relevant.
• The model assumes that probability of relevance depends on the query and the document representations only.
• Ideal answer set is referred to as $R$.
• Documents in the set $R$ are predicted to be relevant.
  – how to compute probabilities?
  – what is the sample space?
The Ranking

Probabilistic ranking computed as:

\[ \text{sim}(q,d_j) = \frac{P(d_j \text{ relevant-to } q)}{P(d_j \text{ non-relevant-to } q)} \]

- How to read this? “Maximize the number of relevant documents, minimize the number of irrelevant documents”
- This is the odds of the document \( d_j \) being relevant

Definition:

- \( w_{ij} \in \{0,1\} \)
- \( P(R \mid d_j) \): probability that document \( d_j \) is relevant
- \( P(\neg R \mid d_j) \): probability that \( d_i \) is not relevant
- Use Bayes Rule: \( P(A\mid B) \ P(B) = P(B\mid A)P(A) \)
The Ranking

\[ \text{sim}(d_j, q) = \frac{P(R \mid d_j)}{P(\neg R \mid d_j)} \]

\[ = \frac{[P(d_j \mid R) \times P(R)]}{[P(d_j \mid \neg R) \times P(\neg R)]} \]

\[ \sim \frac{P(d_j \mid R)}{P(d_j \mid \neg R)} \]

- \( P(d_j \mid R) \): probability of randomly selecting the document \( d_j \) from the set \( R \) of relevant documents.
- Note that \( P(R) \) and \( P(\neg R) \) are the same for all documents in the collection for the given query.
The Ranking

\[ \text{sim}(d_j, q) \sim \frac{P(d_j | R)}{P(d_j | \neg R)} \]
\[ \sim \left[ \prod P(k_i | R) \right] \times \left[ \prod P(\neg k_i | R) \right] \]
\[ \frac{\left[ \prod P(k_i | \neg R) \right] \times \left[ \prod P(\neg k_i | \neg R) \right]}{\left[ \prod P(k_i | \neg R) \right] \times \left[ \prod P(\neg k_i | \neg R) \right]} \]

\[ P(k_i | R) : \text{probability that the index term } k_i \text{ is present in a document randomly selected from the set } R \text{ of relevant documents} \]

- Based on independence assumption
  - Strong assumption!
    - In real life, does not always hold
The Ranking

\[ \text{sim}(d_j, q) \sim \log \left[ \prod P(k_i | R) \right] \ast \left[ \prod P(\neg k_i | R) \right] \]

\[ \sim \left[ \log \prod \frac{P(k_i | R)}{P(\neg k_i | R)} \right] + \left[ \log \prod \frac{P(k_i | \neg R)}{P(\neg k_i | \neg R)} \right] \]

\[ \sim \sum w_{iq} \ast w_{ij} \ast \left( \log \frac{P(k_i | R)}{P(\neg k_i | R)} + \log \frac{P(k_i | \neg R)}{P(\neg k_i | \neg R)} \right) \]

where \[ P(\neg k_i | R) = 1 - P(k_i | R) \]
\[ P(\neg k_i | \neg R) = 1 - P(k_i | \neg R) \]
The Initial Ranking

- $\text{sim}(d_j, q) \sim \sum w_{iq} \cdot w_{ij} \cdot (\log \frac{P(k_i | R)}{P(\neg k_i | R)} + \log \frac{P(k_i | \neg R)}{P(\neg k_i | \neg R)})$

- Probabilities $P(k_i | R)$ and $P(k_i | \neg R)$?

- Estimates based on assumptions:
  - $P(k_i | R) = 0.5$
  - $P(k_i | \neg R) = \frac{n_i}{N}$
    where $n_i$ is the number of docs that contain $k_i$
  - Use this initial guess to retrieve an initial ranking
  - Improve upon this initial ranking
Improving the Initial Ranking

- \( \text{sim}(d_j, q) \sim \sum w_{iq} * w_{ij} * (\log \frac{P(k_i | R)}{P(\neg k_i | R)} + \log \frac{P(k_i | \neg R)}{P(\neg k_i | \neg R)}) \)

- \( V \): set of docs initially retrieved
- \( V_i \): subset of docs retrieved that contain \( k_i \)

- Reevaluate estimates:
  - \( P(k_i | R) = \frac{V_i}{V} \)
  - \( P(k_i | \neg R) = \frac{n_i - V_i}{N - V} \)

- Repeat recursively
Improving the Initial Ranking

• \( \text{sim}(d_j,q) \sim \sum w_{iq} \times w_{ij} \times \left( \log \frac{P(k_i | R)}{P(\neg k_i | R)} + \log \frac{P(k_i | \neg R)}{P(\neg k_i | \neg R)} \right) \)

• To avoid problems with \( V=1 \) and \( V_i=0 \):
  - \( P(k_i | R) = \frac{V_i + n_i/N}{V + 1} \)
  - \( P(k_i | \neg R) = \frac{n_i - V_i + n_i/N}{N - V + 1} \)
  - (replace \( n_i/N \) with 0.5)
Okapi Formula (BM25) (Robertson and Sparck-Jones, 1976)

\[ w_{i,j} = \frac{tf_{i,j} \log \left( \frac{N - df_i + 0.5}{df_i + 0.5} \right)}{k_1 \times ((1 - b) + b \frac{dl}{avdl}) + tf_{i,j}} \]

- \( N \) = number of documents in the collection
- \( tf_{i,j} \) = frequency of term i id document j
- \( df_i \) = number of documents that contain term j
- \( dl \) = length of document j
- \( avdl \) = average length over documents
- \( k1 \) and \( b \) are parameters

- Use this weight in VSM or plug in the probabilistic formula.
Latent Semantic Indexing (LSI)

• **Approach:** Treat word-to-document association data as an unreliable estimate of a larger set of applicable words lying on ‘latent’ dimensions.

• **Goal:** Cluster similar documents which may share no terms in a low-dimensional subspace (improve recall).

• **Preprocessing:** Compute low-rank approximation to the original term-by-document (sparse) matrix

• **Vector Space Model:** Encode terms and documents using factors derived from SVD

• **Evaluation:** Rank similarity of terms and docs to query via Euclidean distances or cosines
Singular Value Decomposition Encoding

- Computes a truncated SVD of the document-term matrix, using the singular vectors as axes of the lower dimensional space
- $A_k$ is the best rank-$k$ approximation to the term-by-document matrix $A$
- Want minimum number of factors ($k$) that discriminate most concepts
- In practice, $k$ ranges between 100 and 300 but could be much larger.
- Choosing optimal $k$ for different collections is challenging.
Strengths and weaknesses of LSI

- ‘Conceptual IR’ recall improvement: one can retrieve relevant documents that do not contain any search terms.
- Calculation of LSI is expensive.
- Continuous normal-distribution-based methods not really appropriate for count data.
- Often improving precision is more important: need query and word sense disambiguation.