Chapter 4: Searching for Text Documents

Ville Turunen

ville.t.turunen@tkk.fi

Introduction

- Multimedia documents usually contain textual parts
- Techniques for text retrieval have been developed in the area of information retrieval (IR)
- Professional users
  - Libraries, archives, etc
  - Complicated Boolean queries
- Novice users
  - Google etc.
  - Natural language queries

Text Documents and Indexing

- Document: list of words and identification
- Indexing: Deriving and storing metadata from documents
- For text documents, *terms* describe the contents
  1. Manually *assigned terms* by professional users
  2. Automatically *derived terms*

Steps in automatic indexing

1. Identify all words and put to lower case
2. Remove *stop words*
    - Words that have little meaning (“the”, “it”...)
3. *Stemming* or *lemmatization*
    - Reduce inflected word forms to their stem:
        - walking, walked $\longrightarrow$ `walk`
    - More complex languages (e.g. Finnish) require more complex algorithms (e.g. *morphological analysis*)
4. Construct *inverted index*
    - References to documents for each term

Query Formulation

- Users need to represent their information need
- Professional searcher knows the document collection and the assigned terms and can use Boolean operators to compose the query
- End user likes to communicate in natural language
  - Derive terms from the query similarly as for the documents (stemming, stop word removal)

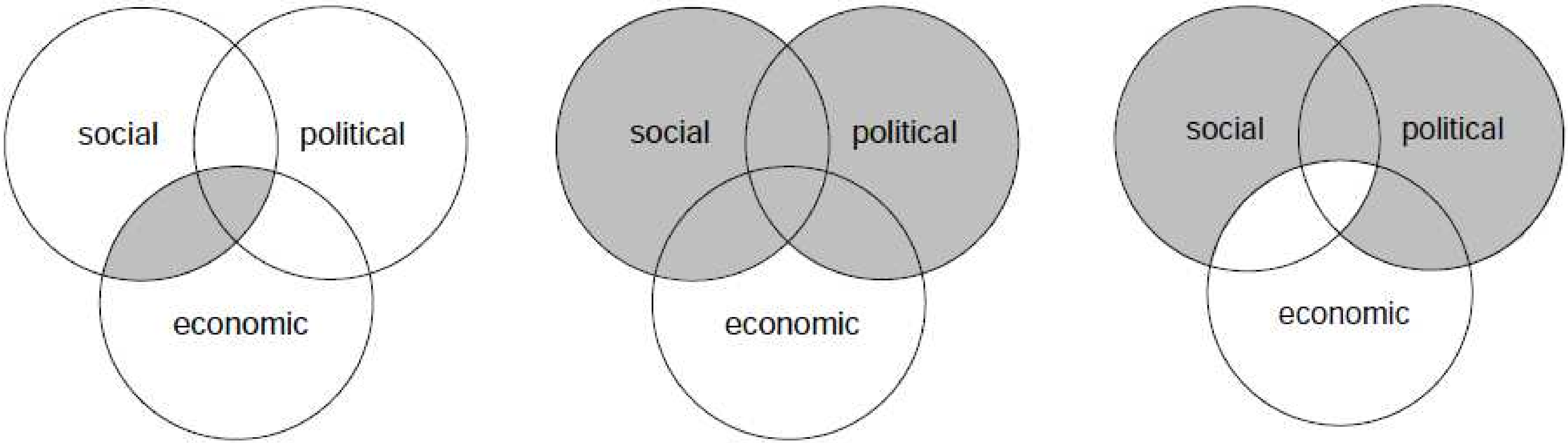
Matching

- Matching algorithm compares the query against the index

1. Exact matching algorithms
    - yes/no decision: the document either matches the query or not
        - Boolean model
2. Inexact matching algorithms
    - System returns a ranked list of documents
    - Relevant documents should be listed first
        - Vector space model
        - Probabilistic model
        - p-norm extended Boolean model
        - Bayesian network model

Boolean Model 1/2

- A query term defines a set of documents
- Terms combined with Boolean operators



social AND economic

social OR political

(social OR political) NOT (social AND economic)

Boolean Model 2/2

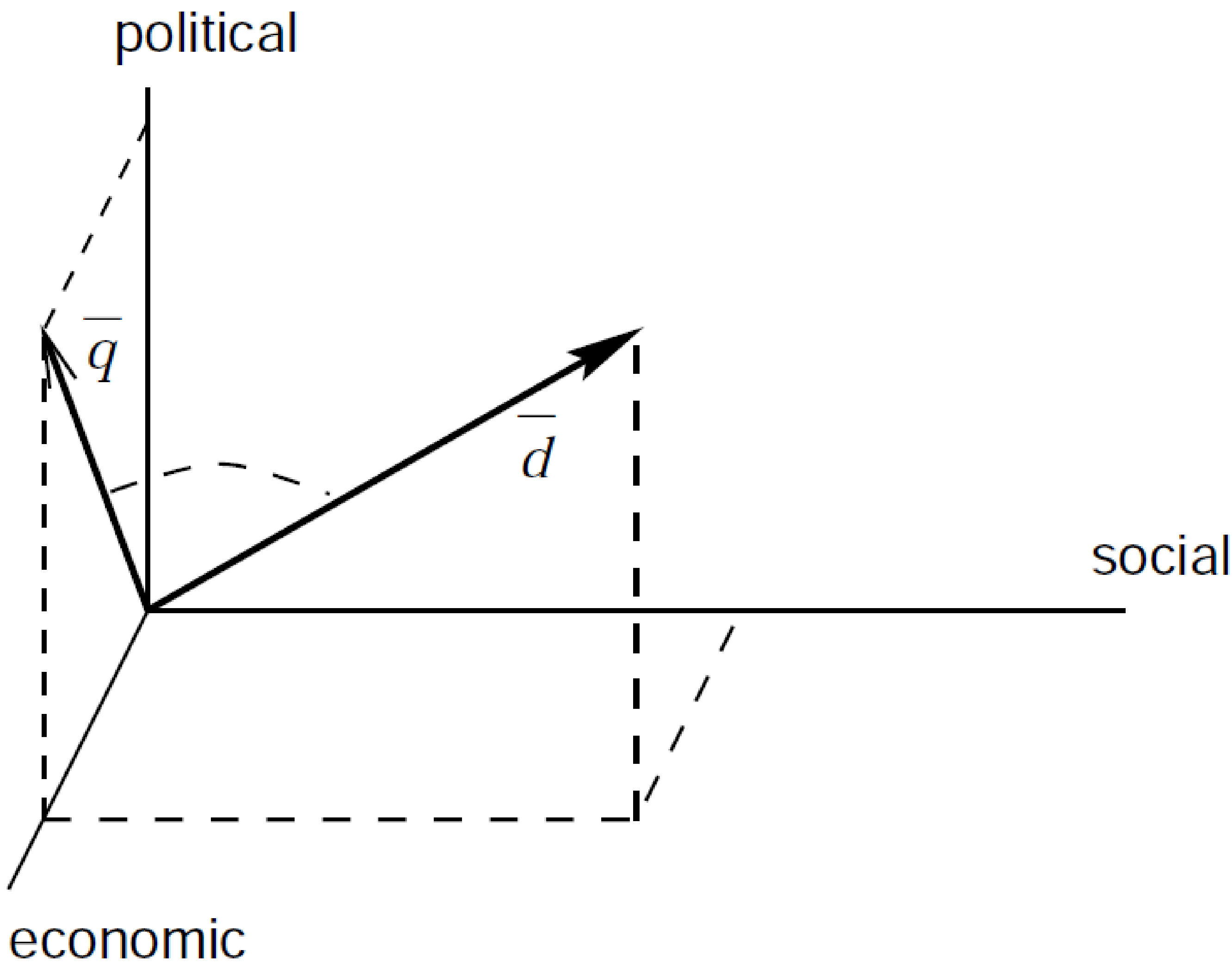
- Proximity searching
  - ADJ: matches if words are adjacent
  - NEAR: matches if words are near each other
- Wildcards
  - Mask part of query: `dog*` matches `dog`, `dogs`, `dogma`
- Pros
  - Very controllable
- Cons
  - Does not rank documents
  - Expert knowledge needed
  - More complex than real needs of users would justify

Vector Space Model 1/2

- Documents are ranked by their *degree of similarity* to the query
- Documents and queries are represented as vectors in high-dimensional Euclidean space
  - document: $\mathbf{d} = (d_1, d_2, \cdots, d_m)$
  - each d_k $(1 \leq k \leq m)$ is associated with an index term
  - similarly for a query: $\mathbf{q} = (q_1, q_2, \cdots, q_m)$
- Similarity measure: cosine of the angle that separates the vectors $\mathbf{d}$ and $\mathbf{q}$:

$$\mathrm{score}(\mathbf{d}, \mathbf{q}) = \frac{\sum_{k=1}^{m} d_k \cdot q_k}{\sqrt{\sum_{k=1}^{m} (d_k)^2} \cdot \sqrt{\sum_{k=1}^{m} (q_k)^2}}$$

Vector Space Model 2/2



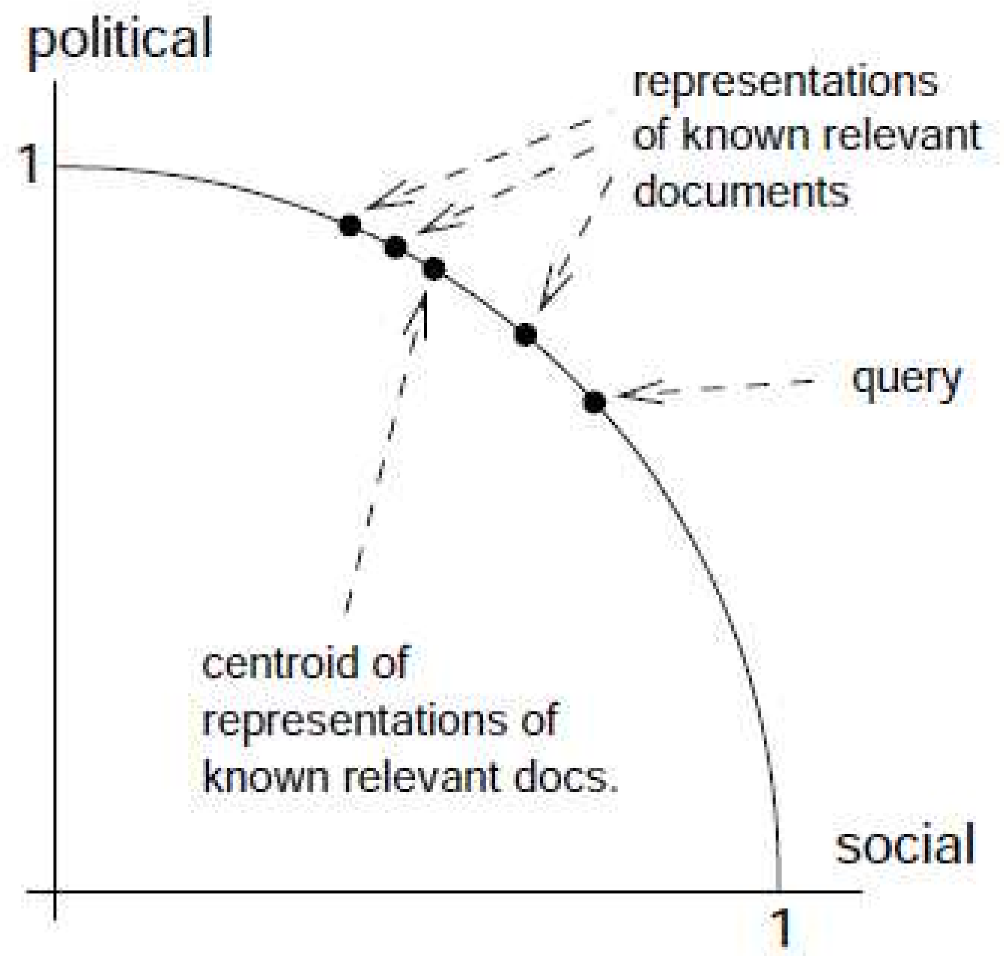
Relevance feedback 1/2

- If relevance of some documents is know (e.g. given by the user), results can be refined
- Move the query vector towards the centroid of the known relevant documents and away from the centroid of known non-relevant documents

$$\mathbf{q}_{new} = \mathbf{q}_{old} + \frac{1}{r}\sum_{i=1}^{r}\mathbf{d}_{rel}^{(i)} - \frac{1}{n}\sum_{i=1}^{n}\mathbf{d}_{nonrel}^{(i)} \tag{1}$$

- $\mathbf{q}_{old}$ is the original query, $\mathbf{q}_{new}$ is the revised query, $\mathbf{d}_{rel}^{(i)}$ is one of the r documents selected as relevant, $\mathbf{d}_{nonrel}^{(i)}$ is one of the n documents selected as non-relevant
- Assumes normalized vectors

Relevance feedback 2/2



Vector Space Model: Discussion

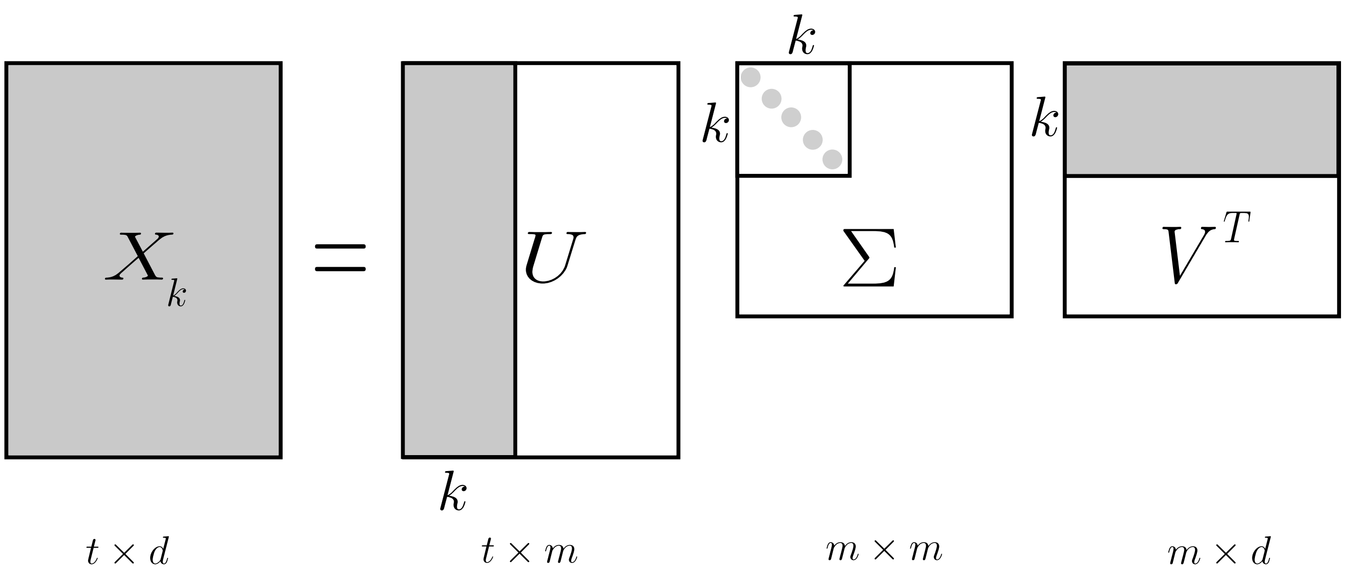
- Pros
  - Intuitive, easily explained
- Cons
  - Does not define what the values of the vector components should be ($\Rightarrow$ term weighting)
  - Not possible to include term dependencies, e.g. phrases or adjacent terms

Term Weighting

- Defines vector component values d_k based on term statistics
- Single most important factor in the performance of IR systems
- *Term frequency*, tf
  - Number of times term occurs within a document
- *Inverse document frequency*, idf
  - Inverse of the number of documents a term occurs in
- $tf.idf$: $d_k = q_k = \mathrm{tf} \cdot \log \frac{N}{\mathrm{df}}$
- Hundreds of variations exist

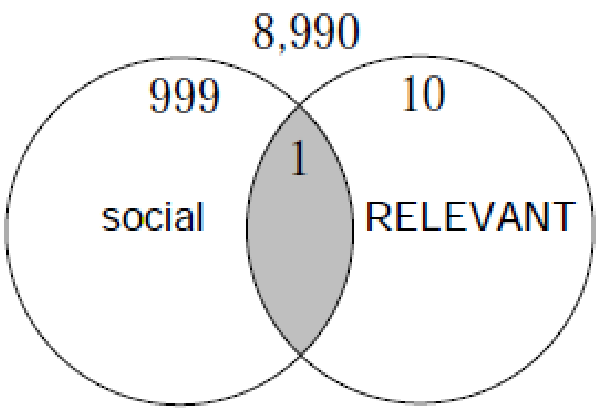
Latent Semantic Indexing

- Arrange document vectors to a *term-document matrix*
- Singular value decomposition is used to project the matrix to fewer dimensions
- These dimensions are hoped to match the "true", latent, meaning of the terms



Probabilistic Model

- Rank the documents in order of their *probability of relevance*
- Motivation: similarity criterion and relevance criterion do not always coincide
- Given query term `social` and known relevances:

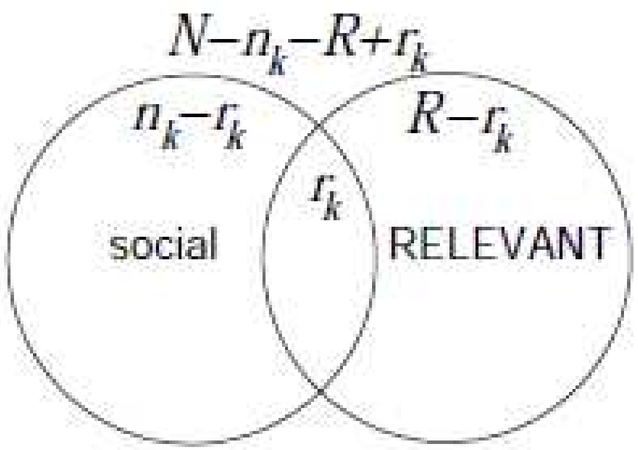


- $P(rel|\texttt{social}) = 1/1000$
- $P(rel|not\ \texttt{social}) = 10/9000$

- In this case, rank by dissimilarity would be optimal

Probability of Relevance 1/2

- Let $L \in \{0, 1\}$ be random variable "document is relevant"
- Let a query contain n terms
- To each document assign n random variables D_k $(1 \leq k \leq n)$ indicating "the document belongs to the subset indexed with kth query term"



$$P(D_k=1|L=1) = r_k/R$$

$$P(D_k=1|L=0) = n_k-r_k/N-R$$

$$P(D_k=0|L=1) = R-r_k/R$$

$$P(D_k=0|L=0) = N-n_k-R+r_k/N-R$$

Probability of Relevance 2/2

- Independence assumption: *In documents terms occur independently from each other*
  - $P(\texttt{social}, \texttt{political} | L = 1) = P(\texttt{social} | L = 1) \cdot P(\texttt{political} | L = 1)$
- Goal is to compute probability that document is relevant given values for $D_1, D_2, \cdots, D_n$.
- Using Bayes rule and the independence assumption, the score for the documents turns out to be:
- $P(L = 1 | D_1, \cdots, D_n) \ \alpha \sum_{k \in \mathrm{m.terms}} \log \frac{P(D_k=1|L=1)P(D_k=0|L=0)}{P(D_k=1|L=0)P(D_k=0|L=1)}$

Probabilistic Model: Discussion

- Pros
  - Does not need additional term weighting
- Cons
  - The distribution of terms over relevant and non-relevant documents is required
    - Needed for $P(D_k|L)$
    - Relevance feedback or assumptions can be used
  - Only defines a partial ranking of the documents i.e. documents in the same non-overlapping subset receive same probability
  - E.g. a short query may return the same rank for first 100 documents

p-norm Extended Boolean Model 1/2

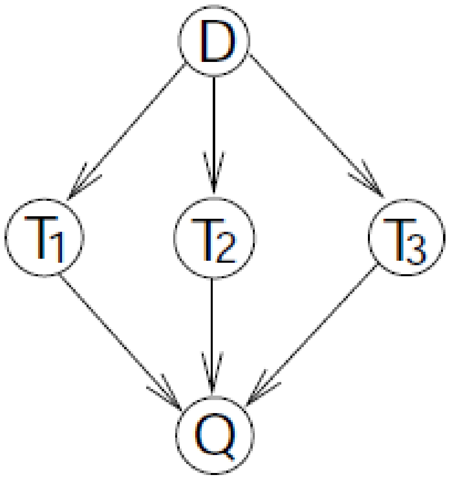
- Uses the idea of documents in vector space
- For two terms:
  - point (1,1): both terms are present
  - point (0,0): both terms are absent
- AND-queries should rank documents in order of increasing distance from point (1,1)
- OR-queries should rank in order of decreasing distance from point (0,0)
  - $\text{score}(\mathbf{d}, a \text{ OR } b) = \sqrt{\frac{(d_a - 0)^2 + (d_b - 0)^2}{2}}$
  - $\text{score}(\mathbf{d}, a \text{ AND } b) = 1 - \sqrt{\frac{(1 - d_a)^2 + (1 - d_b)^2}{2}}$

p-norm Extended Boolean Model 2/2

- Use p-norm instead of Euclidean
- Use weights for query terms
  - $\mathrm{score}(\mathbf{d}, \mathbf{q}_{\mathrm{OR}_{(p)}}) = \left(\frac{\sum_{k=1}^{m}(q_k)^p(d_k)^p}{\sum_{k=1}^{m}(q_k)^p}\right)^{1/p}$
  - $\mathrm{score}(\mathbf{d}, \mathbf{q}_{\mathrm{AND}_{(p)}}) = 1 - \left(\frac{\sum_{k=1}^{m}(q_k)^p(1-d_k)^p}{\sum_{k=1}^{m}(q_k)^p}\right)^{1/p}$
- Pros
  - Performs well
- Cons
  - Needs additional term weighting

Bayesian Network Models

- Bayesian network is an acyclic directed graph that encodes probabilistic dependency relationships
- Nodes are random variables, arrows indicate dependency



- $D = 1$ means document is relevant
- T_1, T_2, T_3 are query terms
- $Q = 1$ means information need is satisfied

- $P(D, T_1, T_2, T_3, Q) =$
  $P(D)P(T_1|D)P(T_2|D)P(T_3|D)P(Q|T_1, T_2, T_3)$

Bayesian Network Models

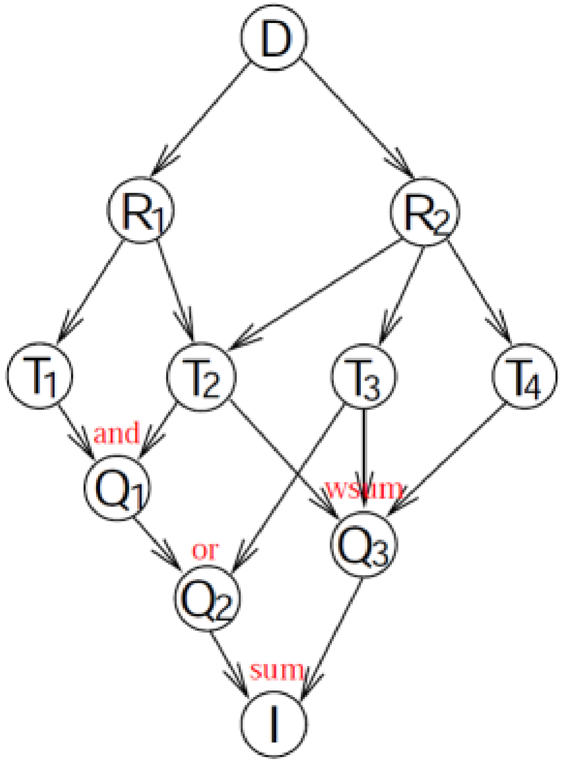
- Rank documents by $P(Q = 1|D = 1)$

$$\begin{aligned} P(Q = 1|D = 1) &= P(Q = 1, D = 1)/P(D = 1) \\ &= \frac{\sum_{t_1,t_2,t_3} P(D = 1, T_1 = t_1, T_2 = t_2, T_3 = t_3, Q = 1)}{P(D = 1)} \end{aligned}$$

- $P(Q|T_1, T_2, \cdots, T_n)$ has 2^{n+1} possible values for a query of length n
- Simplification: use canonical forms
- Suppose $P(T_1|D) = p_1$,$P(T_2|D) = p_2$ and $P(T_3|D) = p_3$ are known

Bayesian Network Models: Canonical forms

$$
\begin{aligned}
P_{and}(Q = 1|D = 1) &= p_1 p_2 p_3 \\
P_{or}(Q = 1|D = 1) &= 1 - (1 - p_1)(1 - p_2)(1 - p_3) \\
P_{sum}(Q = 1|D = 1) &= (p_1 + p_2 + p_3)/3 \\
P_{wsum}(Q = 1|D = 1) &= w_1 p_1 + w_2 p_2 + w_3 p_3
\end{aligned}
$$



- R_1, R_2 different representations for D
- Q_1, Q_2, Q_3 different queries for same need I
- e.g. Q_2 is evaluated as $\mathtt{or}(\mathtt{and}(T_1, T_2)T_3)$ and Q_3 as $\mathtt{wsum}(T_2, T_3, T_4)$

Bayesian Network Models: Discussion

- Pros
  - Network topology can be used to combine evidence in a complex way
- Cons
  - $P(T_i|D)$ need to be estimated
  - Calculation of probabilities take exponential time, if canonical forms not used
  - However, approximation has same effect as changing topology
  - Updating probabilities still intractable

Language Model 1/4

- Language model is a mathematical model of language
- E.g. list of words and their frequencies
- Language modeling studied extensively for automatic speech recognition
- For retrieval:
  - Build a language model for each document
  - Rank documents by probability that the language model of each document generated the query

Language Model 2/4: Urn metaphor

- Someone selects one document
- Draws at random ten words from this document (=query terms)
- Hands those ten words to the system
- System infers from which document the words came from
  - Calculate for each document the probability that the ten words were sampled from it
  - Rank accordingly
- Some query terms may not occur in any relevant docs
  - Before drawing a word, decide randomly whether to draw from a relevant doc or the entire collection
  - Called smoothing the language model distribution

Language Model 3/4

- The probability that a query $T_1, T_2, \cdots, T_n$ is sampled from D:
  - $P(T_1, T_2, \cdots, T_n|D) = \prod_{i=1}^{N}((1-\lambda_i)P(T_i) + \lambda_i P(T_i|D))$
  - λ_i is the relevance weight
- Rank documents by:
  - $P(D|T_1, T_2, \cdots, T_n) = \frac{P(T_1,T_2,\cdots,T_n|D)P(D)}{P(T_1,T_2,\cdots,T_n)}$
- $P(T_1, T_2, \cdots, T_n)$ same for all docs, omitted
- Prior $P(D)$ might be assumed uniform or proportional to length

Language Model 4/4

- Term frequency $tf(t,d)$ and document frequency $df(t)$ can be used to estimate $P(T)$ and $P(T|D)$
  - $P(T_i = t_i | D = d) = \frac{tf(t_i,d)}{\sum_t tf(t,d)}$
  - $P(T_i = t_i) = \frac{df(t_i)}{\sum_t df(t)}$
  - or: $P(T_i = t_i) = \frac{\sum_d tf(t_i,d)}{\sum_d \sum_t tf(t,d)}$
- Language model approach gives theoretical backup for using $tf.idf$ weighting

PageRank in Google (1/2)

- Focus on high quality results instead of similarity
- Pages that have lots of links pointing to them are more important
- Select pages that contain all query terms (Boolean AND)
- Matching pages are ranked by their PageRank
  - $PR(A) = (1-d) + d\left(\frac{PR(T_1)}{C(T_1)} + \cdots + \frac{PR(T_n)}{C(T_n)}\right)$,
  - $PR(A)$ is PageRank of page A, $PR(T_1)$ is PageRank of page T_1, $C(T_i)$ is the number of outgoing links from page T_i and d is a damping factor
  - Recursive

PageRank in Google (2/2)

- Motivation: random surfer model
  - Random surfer visits a page with probability derived from PR
  - Surfer randomly selects one link
  - Or: with probability $(1-d)/N$ surfer gets bored and jumps to another random page

References

[1] H.M Blanken, A.P. de Vries, A.P., H.E. Blok, and L. Feng (Eds.). Multimedia Retrieval. Springer 2007.

[2] Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science*, 41(6):391–407, 1990.

[3] N. Fuhr. Probabilistic Models in Information Retrieval. *The Computer Journal*, 35(3):243–255, 1992.

[4] Djoerd Hiemstra. Using language models for information retrieval, Ph.D. thesis University of Twente, 2001.