Data Mining and Machine Learning

Madhavan Mukund

Lecture 21, Jan–Apr 2020
https://www.cmi.ac.in/~madhavan/courses/dmml2020jan/

Information Retrieval (IR)

- Query a corpus of text documents
- Requirement is an "information need"
- Articulate as a "query", using some fixed syntax
  - How effectively does the query capture the requirement?
- Response is a (ranked) list of documents from the corpus
  - Does this response answer the information need?
- IR traditionally used to effectively index published material
  - Library cataloguing
  - Law: index case histories to look up legal precedents
- Modern context: web search
  - Corpus is all webpages on internet
  - Query is free text in a search box

Information Retrieval (IR)

- Preprocessing for quick response
- Traditional IR focussed on indexing metadata
  - Infeasible to index contents manually

Library of Congress Cataloging-in-Publication Data
Kozen, Dexter, 1951–
Automata and computability/Dexter C. Kozen.
p. cm. – (Undergraduate texts in computer science)
Includes bibliographical references and index.
ISBN 0-387-94907-0 (hardcover: alk. paper)
1. Machine theory. 2. Computable functions. I. Title.
II. Series.
QA267.K69 1997
511.3–dc21 96-37409

- With electronic documents we can index content
  - Maintain data structures that relate query terms to documents

Term-Document matrix

- Recall, set of words document model
  - Vocabulary V, terms of interest
    - "Terms" include words, but also part numbers, proper names, ...
  - Each document d is a subset of V
- Term-document matrix TD
  - Rows are terms, columns are documents
  - $TD[i,j] = 1$ if term i occurs in document j

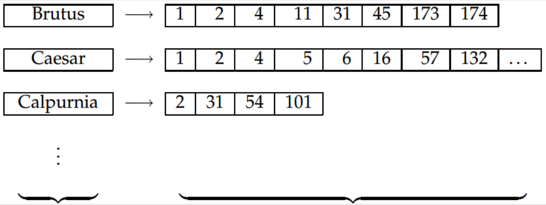
	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
Antony	1	1	0	0	0	1	
Brutus	1	1	0	1	0	0	
Caesar	1	1	0	1	1	1	
Calpurnia	0	1	0	0	0	0	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	0	
...							

Querying a term-document matrix

- Assume a query is a list of key words
- For each query word w, return documents marked 1 in the row for w
- Reduced term-document matrix
  - Retain rows for words in query
  - Retain columns (documents) where at least one query word has an entry 1
- Can intepret list of query words as conjunction or disjunction
  - Conjunction: Return intersection of document lists for individual words
  - Disjunction: Return union of document lists for individual words
  - Perform bitwise and/or down each column (document)
- Answer all boolean queries
  - Negation is also conceptually easy — complement each entry

Postings lists

- Term-document matrix is sparse — most entries are 0
  - Even more so if documents are webpages — typically 2000 words or less in all, most words in V are not present
- Collapse information using inverted index — postings list
  - Each document has a unique ID
  - Each term is linked to list of documents where it occurs, in sorted order of IDs



Vocabulary **Postings**

- Adjacency list vs adjacency matrix representation of a sparse graph

Manipulating postings lists

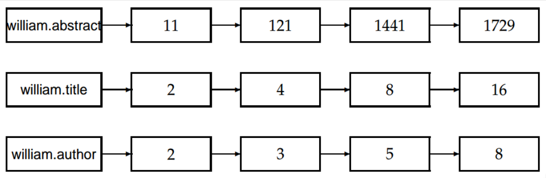
- Each posting list is a sorted list
- Can *merge* two sorted lists into a single sorted list in one pass — union of the lists
- Variations on merge
  - Intersection of the two lists
  - List difference – items in first list but not in second list
- Query is $w_1 w_2$
  - Documents that contain both w_1 and w_2 — intersection merge
  - Documents that contain both w_1 or w_2 — union merge
- Negation is expensive
- Relative complement more useful, corresponds to list difference
  - Documents that contain w_1 but do not contain w_2

Controlling the vocabulary

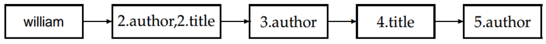
- Remove stop words
  - Common words like *the, and, is, ...*
  - Occur in most documents, not useful to distinguish
  - Limit the size of the vocabulary to reduce postings lists
- Web search engines prefer to retain stop words
  - Computational cost can be managed
  - Useful to match phrases with stop words — *"To be or not to be"*
- Normalization — merging variants of a word to common form
  - Stemming — syntactic, chop down a word to substring
    - Replace, replacing, replacement $\mapsto$ *replac*
  - Lemmatization — semantic, represent words by root form
    - Is, are, were, ... $\mapsto$ *be*

Ranked retrieval

- Search engines return documents ranked by relevance
  - Google's main innovation was an effective ranking mechanism
- Postings lists can only give us a set of unranked documents
- Need extra information to rank
- Zones of a document — title, author, abstract, body, . . .
  - Parametric (zone) index — separate postings lists for each zone

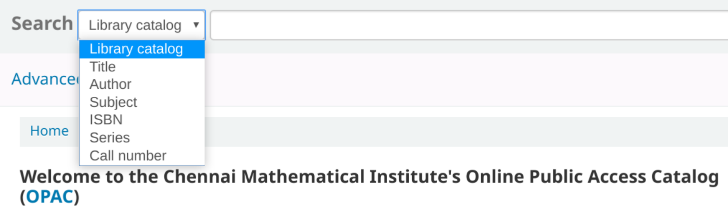


  - Merge zone indices by adding zone information to posting



Ranked retrieval . . .

- Query interface may allow query by zone



- Use weighted zone score to rank responses
  - Zones $i \in \{1, 2, \ldots, k\}$, $s_i = 1$ if term appears in zone i, 0 otherwise
  - Return weighted sum $\sum_{i=1}^{k} g_i s_i$
  - Learn weights g_i using regression — manually labelled data of relevant responses to queries

Beyond the boolean (set-of-words) document model

- Frequency of occurrence of term t in document d is also important
  - Higher frequency indicates more relevance
- Term frequency : $tf_{t,d}$ — how often t occurs in d
- Terms that occur in many documents are not useful (stop words)
  - Term t occurs in n_t documents out of N
  - Usefulness of t is inversely proportional n_t/N
- Inverse document frequency : $idf_t = \log(N/n_t)$
- TF-IDF score of t wrt $d = tf_{t,d} \cdot idf_t$

TF-IDF scores

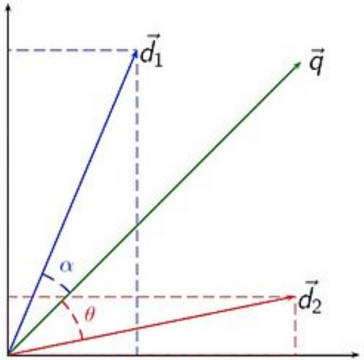
- Postings now record TF-IDF scores
  - $t \rightarrow \{d_1 : tf_{t,d_1} \cdot idf_t, d_2 : tf_{t,d_2} \cdot idf_t, \ldots\}$
- idf_t is independent of document, so factor out of postings, each posting only has $tf_{t,d}$
  - $t \rightarrow \boxed{idf_t}\ idf_t\ , \{d_1 : \boxed{tf_{t,d_1}}\ tf_{t,d_1}\ , d_2 : tf_{t,d_2}, \ldots\}$
  - Compute TF-IDF score by multiplying idf_t, tf_{t,d_j}
- What if we duplicate the content?
  - Copy-pasting content 1000 times boosts TF-score by 1000!
- Traditional IR
  - Books published after editing, review — trustworthy content
- IR for Internet
  - Internet documents are self-published, unverified
  - Economic incentive to boost rankings through fraudulent means
  - Ranking algorithms should try not to be fooled

Vector space model

- Each document is a vector over terms — component i is TF-IDF score for term t_i
- Compare documents in terms of direction
  - $d_1 \cdot d_2 = |d_1||d_2| \cos\theta$
  - $\cos\theta = \dfrac{d_1 \cdot d_2}{|d_1||d_2|}$ measures similarity
- Direction is unaffected by duplication of content — only magnitude changes
  - If d_2 is 1000 copies of d_1, $\cos\theta = 1$
- Search engine can aggregate query responses
  - Collapse similar documents as *"... (5 more like these)"*

Queries in the vector space model

- Treat the query q as a very short document
- For each document d_i, compute $\cos\theta_i$ between q and d_i
- Rank by value of $\cos\theta_i$



Summary

- Precompute term-document information to answer IR queries in set-of-words document model
  - Postings lists are more compact than term-document matrix
- Boolean queries are easy to answer using this information
- Vocabulary can be controlled using stop words, stemming, lemmatization
- Can use weighted zone index for ranked retrieval
- TF-IDF model allows us to account for word frequencies
- Vector space model — cosine similarity
  - Treat query and document as vectors and compare alignment
  - Can also detect similar documents, group related responses