Natural Language Processing

Paper Code: CS-821/6
UG 8th Sem

Dr. Samit Biswas, *Assistant Professor*
Department of Computer Sc. and Technology
Indian Institute of Engineering Science and Technology, Shibpur

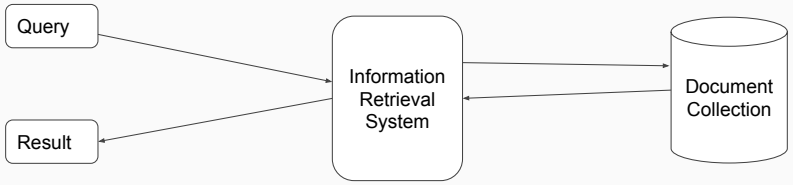
Email: samit@cs.iiests.ac.in

Plan for Today

- Information Retrieval

The Problem of IR

- **Goal:** find documents relevant to an information need from a large document set

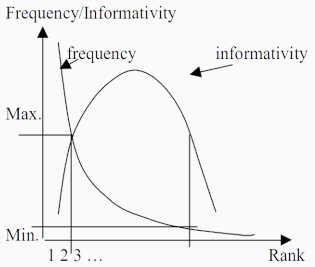


Possible Approaches

- **String matching (Linear search in documents):**
  - *Slow*
  - *Difficult to Improve*
- **Indexing:**
  - *Fast*
  - *Flexible for Further improvement*

Keword Selection and Weighting

- How to select important keywords?
  - Simple method: using middle-frequency words



tf*idf weighting Schema

- **tf = term frequency**
  - frequency of a **term/keyword** in a document
  - The higher the ***tf***, the higher the importance (weight) for the doc.
- **df = document frequency**
  - no. of documents containing the term
  - distribution of the term
- **idf = inverse document frequency**
  - the unevenness of term distribution in the corpus
  - the specificity of term to a document
  - The more the term is distributed evenly, the less it is specific to a document

weight(t, D) = tf(t, D) * idf(t)

Stop words / Stop List

- function words do not bear useful information for IR

  of, in, about, with, I, although, …

- **Stoplist:** contain stopwords, not to be used as index
  - Prepositions
  - Articles
  - Pronouns
  - Some adverbs and adjectives
  - Some frequent words (e.g. document)
- The removal of stopwords usually improves IR effectiveness
- A few “standard” stoplists are commonly used.

Stemming

- Reason:
  - Different word forms may bear similar meaning (e.g. search, searching): create a “standard” representation for them
- Stemming:
  Removing some endings of word

  computer
  compute
  computes
  computing
  computed
  computation

  comput

Result of Indexing

- Each document is represented by a set of weighted keywords (terms):
  D1 → {(t1, w1), (t2,w2), …}

  e.g. D1 → {(comput, 0.2), (architect, 0.3), …}
  D2 → {(comput, 0.1), (network, 0.5), …}
- **Inverted file:**
  comput → {(D1,0.2), (D2,0.1), …}
  Inverted file is used during retrieval for higher efficiency.

Retrieval

- The problems underlying retrieval
  - Retrieval model
    - How is a document represented with the selected keywords?
    - How are document and query representations compared to calculate a score?
- Implementation

Cases

- **1-word query:**
  - The documents to be retrieved are those that include the word
    - Retrieve the inverted list for the word
    - Sort in decreasing order of the weight of the word
- **Multi-word query?**
  - Combining several lists
  - How to interpret the weight?
    (IR model)

IR Model

- Matching Score Model
- Boolean Model
- Extension to Boolean Model
- Vector Space Model

IR Model ...

Matching Score Model

- Document D = a set of weighted keywords
- Query Q = a set of non-weighted keywords
- $R(D, Q) = \Sigma_i w(t_i, D)$
  where t_i is in Q.

Boolean Model

- Document D = Logical Conjunction of keywords
- Query Q = Boolean Expression of keywords (using AND, OR and NOT)
- $R(D, Q) = D \rightarrow Q$
  - e.g. $D = t_1 \wedge t_2 \wedge \ldots \wedge t_n$
    $Q = (t_1 \wedge t_2) \vee (t_3 \wedge \neg t_4)$
    $D \rightarrow Q$, thus $R(D, Q) = 1$.

Example:

- **Document Representation:** Documents are viewed as a set of terms.
  - Document D_1 : **Big cats are nice and funny**
  - Set representation of D_1 after normalization (Tokenization, Stemming and removal of stop words)
    D_1 = **{big, cat, nice, funny}**
- **Query Representation:** Query contains terms and boolean operators **AND, OR**, and **NOT**
  - Example Query: Retrieve all Documents with funny and dog
  - Boolean Expression of the Query:
    funny **AND** dog = funny $\wedge$ dog

Methodology

- Construct the **term - document** incidence matrix
- Apply the following rules:
  - If **Query =** $t_x \wedge t_y$ Documents with both t_x and t_y will be retrieved.
  - If **Query =** $t_x \vee t_y$ Documents with either t_x or t_y will be retrieved.
  - If **Query =** $\neg t_x$ Documents without t_x will be retrieved.

Here, t_x and t_y are terms.

Example:

- **Document Corpus:**
  - d_1 = Big cats are nice and funny
  - d_2 = small dogs are better than big dogs
  - d_3 = small cats are afraid of small dogs
  - d_4 = big cats are not afraid of small dogs
  - d_5 =funny cats are not afraid of small dogs
- **Query:**
  - Retrieve all documents with funny and dog
  - Retrieve all documents with big and dog and not funny
- **Question:** Find the relevant documents using Boolean Model

- **Term - Document matrix**

	d_1	d_2	d_3	d_4	d_5
big	1	1	0	1	0
cat	1	0	1	1	1
nice	1	0	0	0	0
funny	1	0	0	0	1
small	0	1	1	1	1
dog	1	1	1	1	1
better	0	1	0	0	0
than	0	1	0	0	0
afraid	0	0	1	1	1
not	0	0	0	1	1

Matrix Element(t, d)
1 If the document in column d contain term in row t.
0 Otherwise

Example:

- **First Query:** funny $\wedge$ dog
  - $D_{funny} = \{d_1, d_5\}$, $D_{dog} = \{d_2, d_3, d_4, d_5\}$
    $D_{funny} \wedge D_{dog} = \{d_5\}$
- **Second Query:** big $\wedge$ dog $\wedge$ (¬funny)
  $D_{big} = \{d_1, d_2, d_4\}$, $D_{dog} = \{d_2, d_3, d_4, d_5\}$
  $D^c_{funny} = \{d_2, d_3, d_4\}$,
  $D_{big} \wedge D_{dog} \wedge (\neg funny) = \{d_2, d_4\}$

Advantages:
- Very efficient and easy to implement
- Predictable and easy to implement
- Structured queries

Disadvantage:
- Exact matching may **retrieve** too few or too many documents.
- Hard to translate a query into a **Boolean** expression.
- All terms are equally weighted, No ranking
- More like data **retrieval** than information **retrieval**.

Related Applications

- Information Filtering
- Finding Page rank
-

References

References

- Daniel Jurafsky & James H. Martin, "*Speech and Language Processing*", Prentice Hall
- Diana Maynard, Kalina Bontcheva, Isabelle Augenstein, "*Natural Language Processing for the Semantic Web*", A Publication in the Morgan & Claypool Publishers series.
- Steven Bird, Ewan Klein and Edward Loper, "*Natural Language Processing with Python*", By O'Reilly.
- Christopher Manning and Hinrich Schtze, "*Foundations of Statistical Natural Language Processing*", The MIT Press.

Thank You

Contacts:

Dr. Samit Biswas
Department of CST,
IIEST, Shibpur

samit@cs.iiests.ac.in
https://www.iiests.ac.in/IIEST/Faculty/cs-samit