Information Retrieval: Basic Models

Mandar Mitra

Indian Statistical Institute

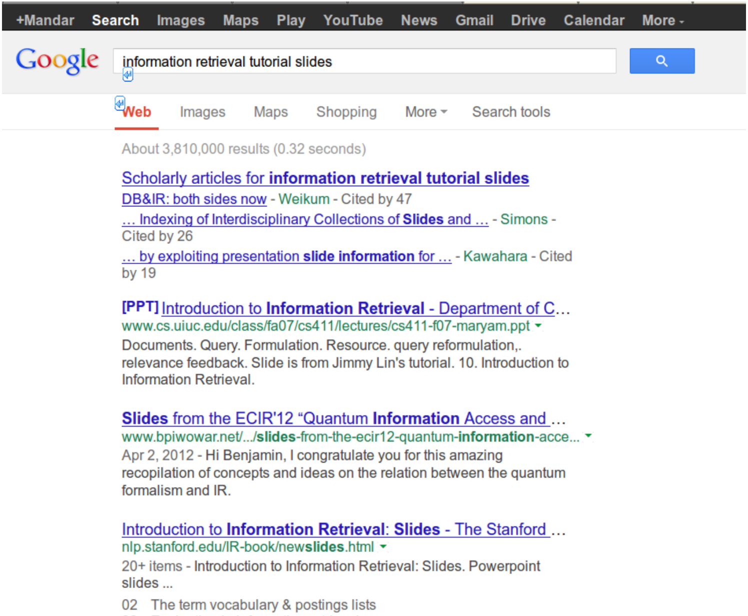
Outline

Information retrieval

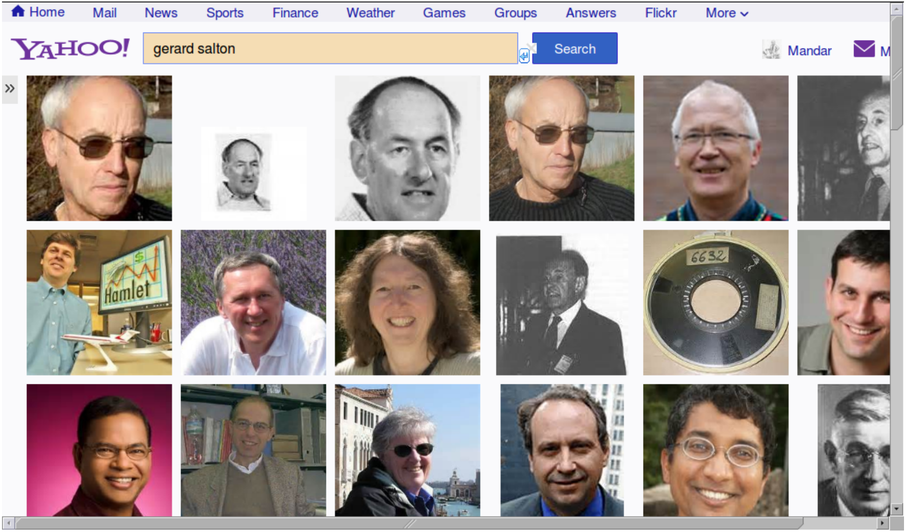
Problem definition:

Given a user's *information need*, find documents satisfying that need.

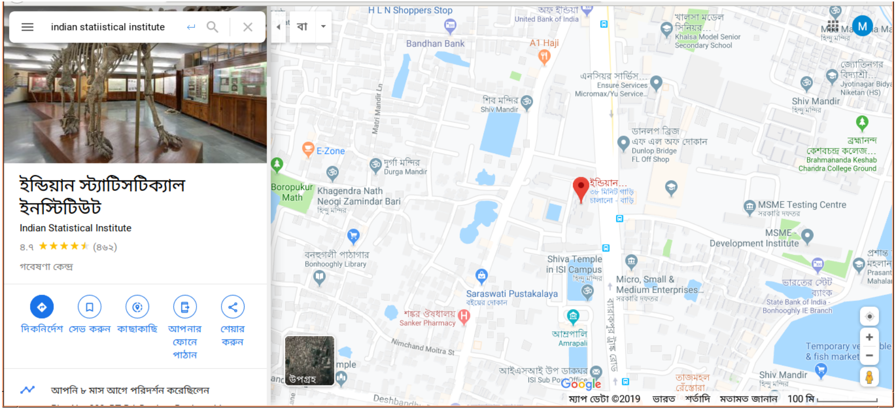
Information retrieval



Information retrieval



Information retrieval



Information retrieval

Problem definition:

Given a user's *information need*, find documents satisfying that need.

Information retrieval

Problem definition:

Given a user's *information need*, find documents satisfying that need.

- Types of information: text, images/graphics, speech, video, etc.
- Text is still the most commonly used.

IR: bag of words approach

- Document $\rightarrow$ list of keywords / content-descriptors / *terms*
- User's information need $\rightarrow$ (natural-language) query $\rightarrow$ list of keywords
- Measure overlap between query and documents.

Indexing

Tokenization: identify individual words.

Information retrieval (IR) is the activity of obtaining information resources relevant to an information need from a collection of information resources. Searches can be based on full-text or other content-based indexing.

$$\Downarrow$$

Information retrieval IR is the activity of obtaining ...

Indexing: tokenization with NLTK

Getting started

```
import nltk
from nltk.book import * # for existing corpora
```

Indexing: tokenization with NLTK

Getting started

```
import nltk
from nltk.book import * # for existing corpora
```

Tokenization I

```
from nltk import word_tokenize
with open('filename.txt') as fp:
    text = fp.read()
    tokenlist = word_tokenize(text)
```

Indexing: tokenization with NLTK

Tokenization II

```
from nltk.corpus import PlaintextCorpusReader
corpus_root = './data'
filelist = PlaintextCorpusReader(corpus_root,
    '.*\.txt')
# filelist.fileids() gives ['file1.txt', 'file2.txt']
# filelist.words('file1.txt') gives [u'Reason',
    u'for', ...
```

Indexing: stopword removal

Eliminate common words

Information retrieval IR is the activity of obtaining . . .

Indexing: stopword removal

Eliminate common words

Information retrieval IR is the activity of obtaining . . .

Stopword removal in NLTK

```
from nltk.corpus import stopwords
stoplist = stopwords.words('english') # [u'i', u'me', u'my', ...
filtered = [ w.lower() for w in filelist.words('file1.txt')
                              if w.isalnum()
                                 and w.lower() not in stoplist ]
```

Indexing: stemming

- Stemming: reduce words to a common root.
    - e.g. resignation, resigned, resigns $\rightarrow$ resign
    - use standard algorithms (Porter).

Indexing: stemming

- Stemming: reduce words to a common root.
  - e.g. resignation, resigned, resigns → resign
  - use standard algorithms (Porter).

Stemming in NLTK

```
porter = nltk.PorterStemmer()
stemmed = [ porter.stem(w) for w in filtered ]
index_terms = sorted(set(stemmed))
```

Indexing: phrases

Phrases: multi-word terms e.g. computer science, data mining.

- Syntactic/linguistic methods
  - use a part of speech tagger
  - look for particular POS sequences, e.g., NN NN, JJ NN
    Example: computer/NN science/NN

Indexing: phrases

- Statistical methods: $f_{(a,b)} > \theta$ (threshold)
  - Raw frequency: $f_{raw}(a,b) = n_{(a,b)}$
  - Dice coefficient:

$$f_{dice}(a,b) = 2 \times n_{(a,b)}/(n_a + n_b)$$

n_a, n_b number of bi-grams whose first (second) word is a (b)

  - Mutual information

$$MI(X,Y) = \sum_{y \in Y} \sum_{x \in X} P(x,y) \log_2 \frac{P(x,y)}{P(x)P(y)}$$

$X_a = 1$ if randomly chosen word $w = a$, 0 otherwise

$$f_{mi}(p) = \sum_{X_a \in \{0,1\}} \sum_{X_b \in \{0,1\}} P(X_a, X_b) \log_2 \frac{P(X_a, X_b)}{P(X_a)P(X_b)}$$

IR: basic principle

- Document $\rightarrow$ list of keywords / content-descriptors / *terms*
- Document collection $\rightarrow$ *Term-Document Matrix*

Vocabulary: set of all words in collection

$$
\text{Document collection} \longrightarrow
\begin{array}{c|c}
 & t_1 \quad t_2 \quad \ldots \quad t_M \\
\hline
\begin{array}{c} D_1 \\ D_2 \\ \vdots \\ D_N \end{array} & N \times M \text{ binary (0-1) matrix}
\end{array}
$$

- User's information need $\rightarrow$ (natural-language) query $\rightarrow$ list of keywords
- Measure overlap between query and documents.

IR: basic principle

- Document $\rightarrow$ list of keywords / content-descriptors / *terms*
- Document collection $\rightarrow$ *Term-Document Matrix*

Vocabulary: set of all words in collection

$$
\text{Document collection} \longrightarrow
\begin{array}{c|c}
 & t_1 \quad t_2 \quad \ldots \quad t_M \\
\hline
\begin{array}{c} D_1 \\ D_2 \\ \vdots \\ D_N \end{array} & N \times M \text{ binary (0-1) matrix}
\end{array}
$$

- User's information need $\rightarrow$ (natural-language) query $\rightarrow$ list of keywords
- Measure overlap between query and documents.

Outline

1 Introduction

2 Models
- Boolean model
- Vector space model
- Language model

Boolean model

- Keywords combined using AND, OR, (AND) NOT

  e.g. (medicine OR treatment) AND (hypertension OR "high blood pressure")

Boolean model

- Keywords combined using AND, OR, (AND) NOT
  e.g. (medicine OR treatment) AND (hypertension OR "high blood pressure")

- Efficient and easy to implement (list merging)
  - AND $\equiv$ intersection
    OR $\equiv$ union
  - Example:
    medicine $\rightarrow D_1, D_4, D_5, D_{10}, \ldots$
    hypertension $\rightarrow D_2, D_4, D_8, D_{10}, \ldots$

Boolean model

- Keywords combined using AND, OR, (AND) NOT
  e.g. (medicine OR treatment) AND (hypertension OR "high blood pressure")

- Efficient and easy to implement (list merging)
  - AND $\equiv$ intersection
    OR $\equiv$ union
  - Example:
    medicine $\rightarrow D_1, D_4, D_5, D_{10}, \ldots$
    hypertension $\rightarrow D_2, D_4, D_8, D_{10}, \ldots$

- Drawbacks
  - OR — one match as good as many
    AND — one miss as bad as all
  - no ranking
  - queries may be difficult to formulate

Vector space model

- Any text item ("document") is represented as list of terms and associated weights.

	t_1	t_2	$\ldots$	t_M
D_1	w_{11}	w_{12}		w_{1M}
D_2	w_{21}	w_{22}		w_{2M}
$\vdots$				
D_N	w_{N1}	w_{N2}		w_{NM}

- Term = keywords or content-descriptors
- Weight = measure of the importance of a term in representing the information contained in the document

Term weights

- Term frequency (tf): repeated words are strongly related to content
  - use sub-linear function
  - examples:

$$1 + \log(tf), \quad 1 + \log\left(1 + \log(tf)\right), \quad \frac{tf}{k + tf}$$

- Inverse document frequency (idf): uncommon term is more important
  Example: medicine vs. antibiotic
  - commonly used functions

$$\log \frac{N}{1 + df}, \quad \log \frac{N - df + 0.5}{df + 0.5}$$

Term weights

- Normalization by document length: term-weights for long documents should be reduced
  - long docs. contain many distinct words.
  - long docs. contain same word many times.
  - Intuition: each term covers a smaller portion of the overall information content of a long document
  - use # bytes, # distinct words, Euclidean length, etc.
- Weight = tf x idf / normalization

Term weights: “traditional” weighting schemes

- Cosine normalisation

$$\frac{(1 + \log(tf)) \times \log \frac{N}{1+df}}{\sqrt{\sum w_i^2}}$$

- Pivoted normalisation

$$\frac{\frac{1+\log(tf)}{1+\log(average\ \ tf)} \quad \times \quad \log(\frac{N}{df})}{(1.0 - slope) \times pivot \quad + \quad slope \times \#\ unique\ \ terms}$$

Term weights: BM25

- Derived from a probabilistic model

$$\frac{tf \quad \times \quad \log(\frac{N - df + 0.5}{df + 0.5})}{k_1((1-b) + b\frac{dl}{avdl}) + tf}$$

Retrieval

- Measure vocabulary overlap between user query and documents.

$$
\begin{array}{rcllll}
 & & t_1 & & \ldots & t_M \\
Q & = & q_1 & & \ldots & q_M \\
D & = & d_1 & & \ldots & d_M \\
Sim(Q, D) & = & \vec{Q}.\vec{D} & & & \\
 & = & \sum_i q_i \times d_i & & &
\end{array}
$$

Retrieval

- Measure vocabulary overlap between user query and documents.

$$
\begin{array}{ccccc}
 & & t_1 & \ldots & t_M \\
Q & = & q_1 & \ldots & q_M \\
D & = & d_1 & \ldots & d_M \\
Sim(Q,D) & = & \vec{Q}.\vec{D} & & \\
 & = & \sum_i q_i \times d_i & &
\end{array}
$$

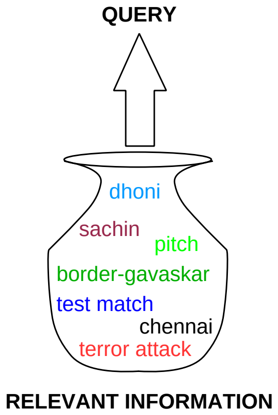
- Use inverted list (index).

$$t_i \rightarrow (D_{i_1}, w_{i_1}), \ldots, (D_{i_k}, w_{i_k})$$

Language model

Basic idea

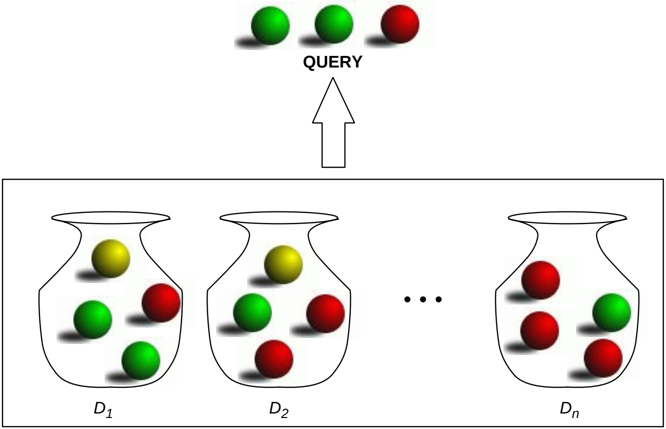
Relevant information = urn = (unigram) probability distribution



Language model

Basic idea

Relevant information = urn = (unigram) probability distribution



Random variables

- D : document (*urn*)
  range = $\{d_1, d_2, \ldots, d_n\}$

- T_i : i-th query term (*coloured balls*)
  range = $\{t_1, t_2, \ldots, t_M\}$

Principle

Which document was my query most likely drawn from?

Rank document by:

$$P(D = d_i | T_1 = t_1, \ldots, T_k = t_k)$$

$$\approx P(D = d_i, T_1 = t_1, \ldots, T_k = t_k)$$

$$= P(T_1 = t_1, \ldots, T_k = t_k | D = d_i) \times P(D = d_i)$$

$$= P(D = d_i) \prod_{j=1}^{k} P(T_j = t_j | D = d_i)$$

Estimating probabilities

$$P(D = d) = \frac{1}{\#\ of\ documents}$$

$$P(T_j = t_j | D = d) = \frac{tf(t_j, d)}{\sum_t tf(t, d)}$$

Rank documents by:

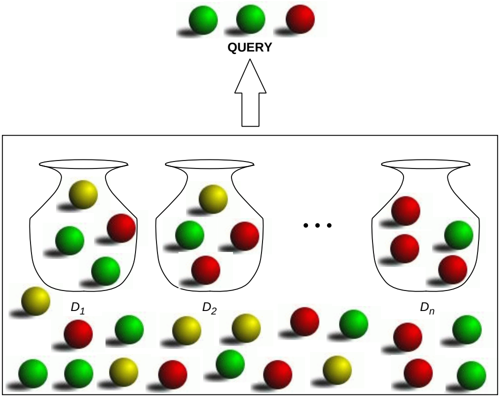
$$P(D = d_i) \prod_{j=1}^{k} P(T_j = t_j | D = d_i)$$

$$\prod_{j=1}^{k} \frac{tf(t_j, d)}{|D|}$$

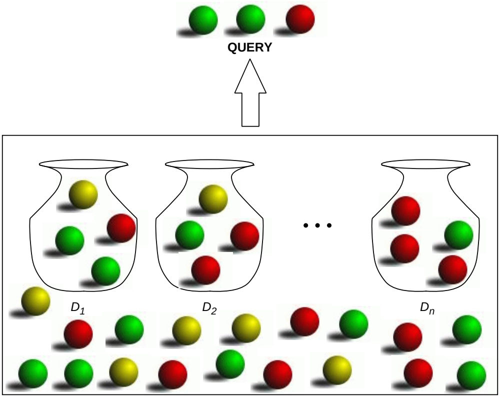
Smoothing

- *Sparse data problem:* training data is only a sample of the entire population
  $\Rightarrow$ possible events may not be observed in training data
- Unseen events assigned *zero* probability during maximum likelihood estimation
- *Smoothing* assigns some non-zero probability to events that were unseen in training data

Smoothing



Smoothing



$$P(T_j = t_j|C) = \frac{\sum_k tf(t_j,k)}{\sum_{t,k} tf(t,k)} = \frac{cf(t_j)}{|C|}$$

Smoothing techniques

- **Linear interpolation / Jelinek-Mercer**

$$P_{JM}(t|d) \;=\; \lambda P_{ML}(t|d) \;+\; (1-\lambda)P_{ML}(t|C)$$

- **Dirichlet** $\left(\lambda = \frac{|D|}{|D|+\mu}\right)$

$$P_D(t|d) = \frac{tf(t,d) + \mu P_{ML}(t|C)}{|D| + \mu}$$

Ranking function

$$P(D = d_i) \prod_{j=1}^{k} P(T_j = t_j | D = d_i)$$

$$= \prod_{j=1}^{k} \left(\lambda \frac{tf(t_j, d)}{|D|} + (1 - \lambda) \frac{cf(t_j)}{|C|} \right)$$

$$\approx \sum_{j=1}^{k} \log \left(1 + \frac{\lambda}{1 - \lambda} \times \frac{tf(t_j, d)}{|D|} \times \frac{|C|}{cf(t_j)} \right)$$

References

- *An Introduction to Information Retrieval.* Manning, Raghavan, Schutze.

  http://www-csli.stanford.edu/~schuetze/information-retrieval-book.html
- *Text Data Management: A Practical Introduction to Information Retrieval and Text Mining.* ChengXiang Zhai and Sean Massung. ACM and Morgan & Claypool Publishers, 2016.
- *The Probabilistic Relevance Framework: BM25 and Beyond.* Robertson, Zaragoza. Foundations and Trends in IR, 3(4), 2009.
- *Using language models for information retrieval.* Djoerd Hiemstra. PhD Thesis, U. Twente. 2000.
- *Text Mining and Analytics.* ChengXiang Zhai.

  https://www.coursera.org/learn/text-mining/home/welcome