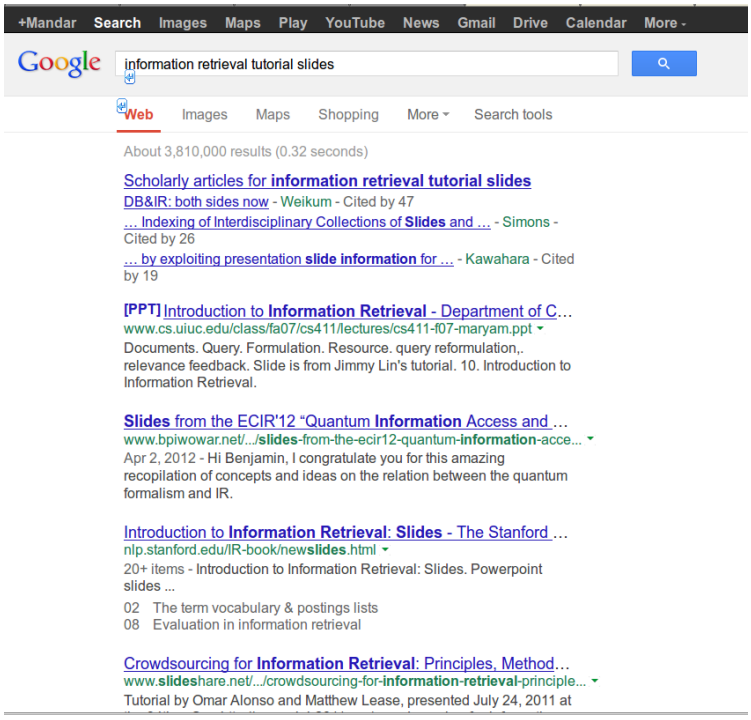
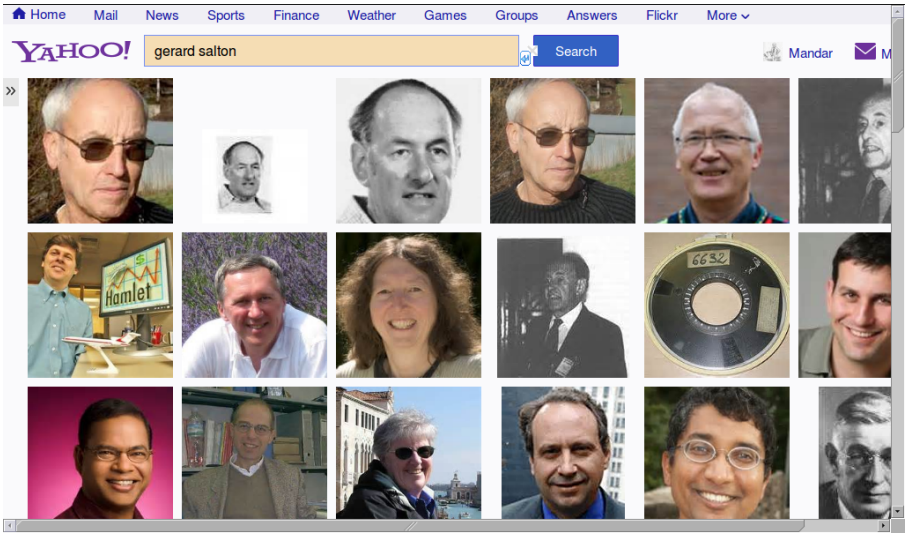
Indexing and Retrieval: Basics

Mandar Mitra

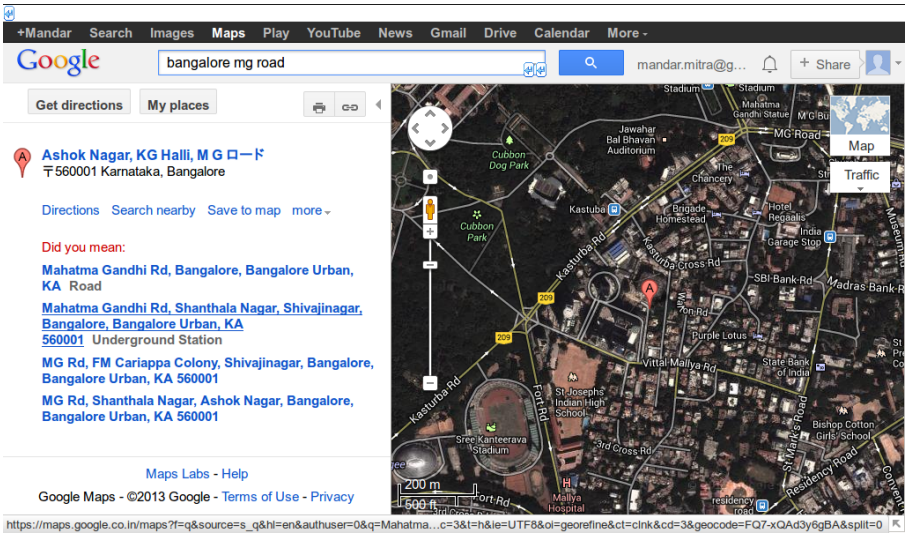
Information retrieval



Information retrieval



Information retrieval



Information retrieval

- *What's the best smartphone under Rs.10,000?*
- *I'm free this evening. How do I entertain myself?*
- …

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 → list of keywords / content-descriptors / *terms*
- User's information need → (natural-language) query → 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.

⇓

Information retrieval IR is the activity of obtaining …

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 ...

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 → resign
  - use standard algorithms (Porter).

Stemming in NLTK

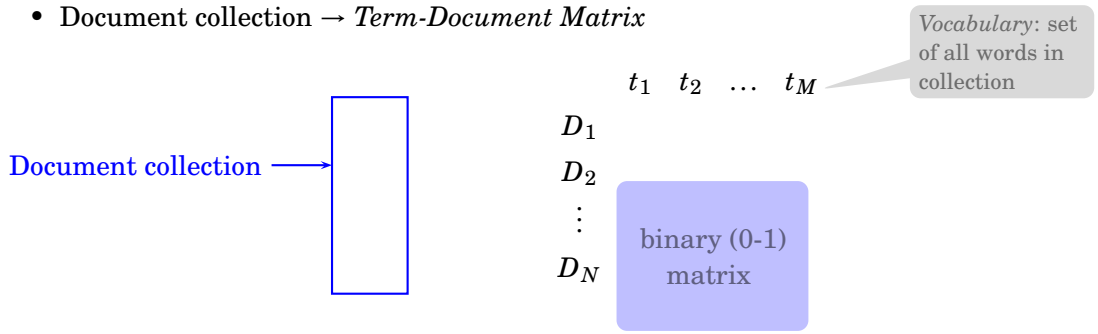
```
porter = nltk.PorterStemmer()
stemmed = [ porter.stem(w) for w in filtered ]
index_terms = sorted(set(stemmed))
```

Indexing

- Thesaurus: find synonyms for words in the document.
- Phrases: find multi-word terms e.g. computer science, data mining.
  - use syntax/linguistic methods or "statistical" methods.
- Named entities: identify names of people, organizations and places; dates; monetary or other amounts, etc.

IR: basic principle

- Document → list of keywords / content-descriptors / *terms*
- Document collection → *Term-Document Matrix*



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

1 Models

1.1 Boolean model

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 ≡ intersection

    OR ≡ union

- Example:
  medicine → $D_1, D_4, D_5, D_{10}, \ldots$ hypertension → $D_2, D_4, D_8, D_{10}, \ldots$

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

1.2 Vector space model

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
- Inverse document frequency (idf): uncommon term is more important
  Example: medicine vs. antibiotic
- Normalization by document length
  - long docs. contain many distinct words.
  - long docs. contain same word many times.
  - term-weights for long documents should be reduced.
  - use # bytes, # distinct words, Euclidean length, etc.
- Weight = tf x idf / normalization

Term weights: commonly used weighting schemes

- Pivoted normalization [Singhal et al., SIGIR 96]

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

- BM25 (probabilistic model) [Robertson and Zaragoza, FTIR 2009]

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

Retrieval

- Measure vocabulary overlap between user query and documents.

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

- Use inverted list (index).

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