Lecture 1: Introduction and the Boolean Model

Information Retrieval
Computer Science Tripos Part II

Simone Teufel

Natural Language and Information Processing (NLIP) Group
UNIVERSITY OF CAMBRIDGE

Simone.Teufel@cl.cam.ac.uk

Overview

What is Information Retrieval?

Manning et al, 2008:

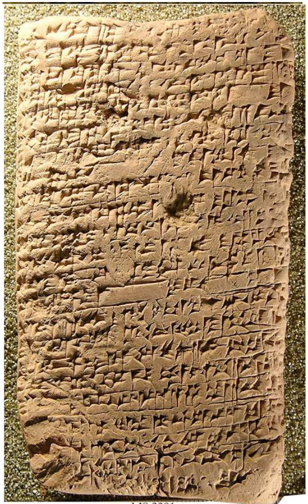
Information retrieval (IR) is finding material . . . of an unstructured nature . . . that satisfies an information need from within large collections . . . .

What is Information Retrieval?

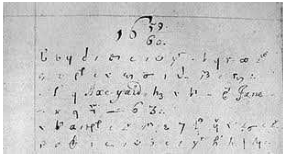
Manning et al, 2008:

Information retrieval (IR) is finding material . . . of an unstructured nature . . . that satisfies an information need from within large collections . . . .

Document Collections


MS 3391
Library catalogue. Babylonia, 2000-1600 BC

Document Collections



IR in the 17th century: Samuel Pepys, the famous English diarist, subject-indexed his treasured 1000+ books library with key words.

Document Collections

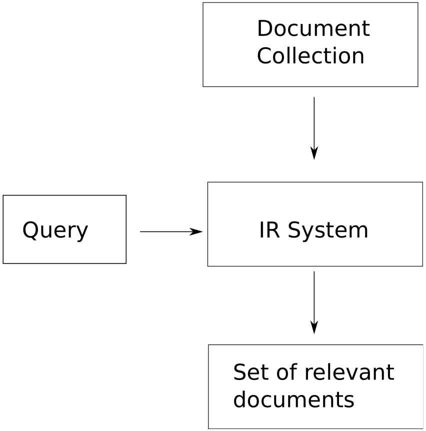
What we mean here by document collections

Manning et al, 2008:

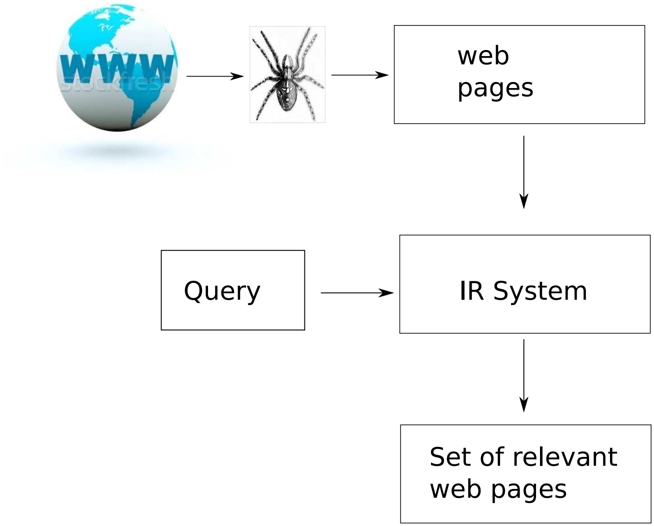
Information retrieval (IR) is finding material (usually documents) of an unstructured nature . . . that satisfies an information need from within large collections (usually stored on computers).

- Document Collection: text units we have built an IR system over.
- Usually documents
- But could be
  - memos
  - book chapters
  - paragraphs
  - scenes of a movie
  - turns in a conversation...
- Lots of them

IR Basics


Document Collection
Query
IR System
Set of relevant documents

IR Basics



What is Information Retrieval?

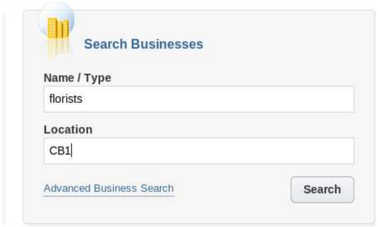
Manning et al, 2008:

Information retrieval (IR) is finding material (usually documents) of an unstructured nature . . . that satisfies an information need from within large collections (usually stored on computers.

Structured vs Unstructured Data

Unstructured data means that a formal, semantically overt, easy-for-computer structure is missing.

- In contrast to the rigidly structured data used in DB style searching (e.g. product inventories, personnel records)



Name / Type: florists

Location: CB1

```
SELECT *
FROM business_catalogue
WHERE category = 'florist'
AND city_zip = 'cb1'
```

- This does not mean that there is no structure in the data
    - Document structure (headings, paragraphs, lists. . .)
    - Explicit markup formatting (e.g. in HTML, XML. . .)
    - Linguistic structure (latent, hidden)

Information Needs and Relevance

Manning et al, 2008:

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

- An information need is the topic about which the user desires to know more about.
- A query is what the user conveys to the computer in an attempt to communicate the information need.
- A document is relevant if the user perceives that it contains information of value with respect to their personal information need.

Types of information needs

Manning et al, 2008:

> Information retrieval (IR) is finding material . . . of an unstructured nature . . . that satisfies an information need from within large collections . . . .

- Known-item search
- Precise information seeking search
- Open-ended search (“topical search”)

Information scarcity vs. information abundance

- Information scarcity problem (or needle-in-haystack problem): hard to find rare information
  - Lord Byron's first words? 3 years old? Long sentence to the nurse in perfect English?

. . . when a servant had spilled an urn of hot coffee over his legs, he replied to the distressed inquiries of the lady of the house, 'Thank you, madam, the agony is somewhat abated.' [not Lord Byron, but Lord Macaulay]

- Information abundance problem (for more clear-cut information needs): redundancy of obvious information
  - What is toxoplasmosis?

Relevance

Manning et al, 2008:

Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

- Are the retrieved documents
  - about the target subject
  - up-to-date?
  - from a trusted source?
  - satisfying the user's needs?
- How should we rank documents in terms of these factors?
- More on this in a lecture soon

How well has the system performed?

The effectiveness of an IR system (i.e., the quality of its search results) is determined by two key statistics about the system's returned results for a query:

- Precision: What fraction of the returned results are relevant to the information need?
- Recall: What fraction of the relevant documents in the collection were returned by the system?
- What is the best balance between the two?
  - Easy to get perfect recall: just retrieve everything
  - Easy to get good precision: retrieve only the most relevant

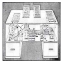
There is much more to say about this – lecture 5

IR today

- Web search (Google bing)
    - Search ground are billions of documents on millions of computers
    - issues: spidering; efficient indexing and search; malicious manipulation to boost search engine rankings
    - Link analysis covered in Lecture 8
- Enterprise and institutional search (PubMed LexisNexis)
    - e.g company's documentation, patents, research articles
    - often domain-specific
    - Centralised storage; dedicated machines for search.
    - Most prevalent IR evaluation scenario: US intelligence analyst's searches
- Personal information retrieval (email, pers. documents;)
    - e.g., Mac OS X Spotlight; Windows' Instant Search
    - Issues: different file types; maintenance-free, lightweight to run in background

A short history of IR

1945

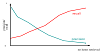


memex

1950s

Term IR coined by Calvin Moers

Literature searching systems; evaluation by P&R (Alan Kent)



1960s

Cranfield experiments

Boolean IR

SMART

1970s 1980s

Salton;

VSM

LexisNexis

MEDLINE

1990s

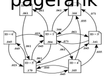
TREC

altavista

YAHOO!

excite

pagerank



2000s

Google

Google Scholar BETA

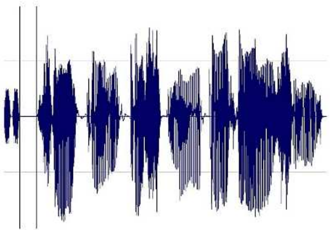
Google news

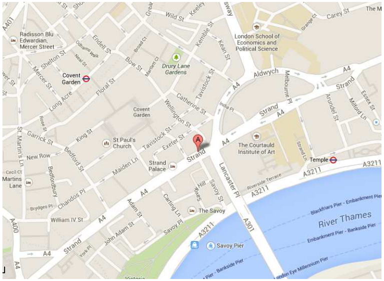
Multimedia

Multilingual (CLEF)

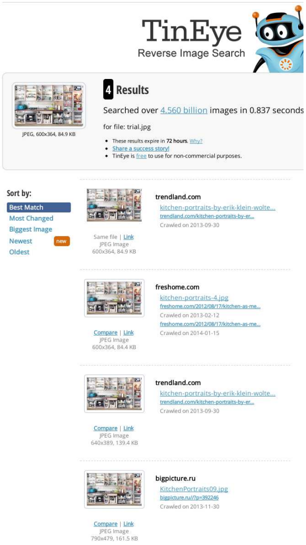
Recommendation Systems

IR for non-textual media





Similarity Searches



TinEye
Reverse Image Search

JPEG, 600x364, 84.9 KB

4 Results

Searched over 4.560 billion images in 0.837 seconds

for file: trial.jpg

- These results expire in 72 hours. Why?
- Share a success story!
- TinEye is free to use for non-commercial purposes.

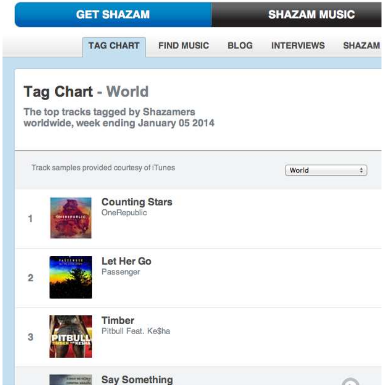
Sort by:
Best Match
Most Changed
Biggest Image
Newest new
Oldest

trendland.com
kitchen-portraits-by-erik-klein-wolte...
trendland.com/kitchen-portraits-by-er...
Crawled on 2013-09-30
Same file | Link
JPEG Image
600x364, 84.9 KB

freshome.com
kitchen-portraits-4.jpg
freshome.com/2012/08/17/kitchen-as-me...
Crawled on 2013-02-12
freshome.com/2012/08/17/kitchen-as-me...
Crawled on 2014-01-15
Compare | Link
JPEG Image
600x364, 84.4 KB

trendland.com
kitchen-portraits-by-erik-klein-wolte...
trendland.com/kitchen-portraits-by-er...
Crawled on 2013-09-30
Compare | Link
JPEG Image
640x389, 139.4 KB

bigpicture.ru
KitchenPortraits09.jpg
bigpicture.ru/?p=392246
Crawled on 2013-11-30
Compare | Link
JPEG Image
790x479, 161.5 KB



GET SHAZAM | SHAZAM MUSIC

TAG CHART | FIND MUSIC | BLOG | INTERVIEWS | SHAZAM

Tag Chart - World

The top tracks tagged by Shazamers worldwide, week ending January 05 2014

Track samples provided courtesy of iTunes — World

1 Counting Stars — OneRepublic
2 Let Her Go — Passenger
3 Timber — Pitbull Feat. Ke$ha
Say Something

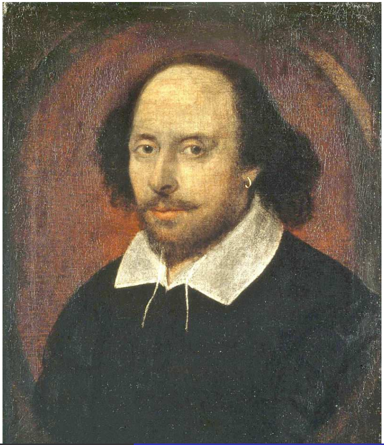
Areas of IR

- “Ad hoc” retrieval (lectures 1-5)
- web retrieval (lecture 8)
- Support for browsing and filtering document collections:
  - Clustering (lecture 6)
  - Classification; using fixed labels (common information needs, age groups, topics; lecture 7)
- Further processing a set of retrieved documents, e.g., by using natural language processing
  - Information extraction
  - Summarisation
  - Question answering

Overview

Boolean Retrieval

- In the Boolean retrieval model we can pose any query in the form of a Boolean expression of terms
- i.e., one in which terms are combined with the operators and, or, and not.
- Shakespeare example



Brutus AND Caesar AND NOT Calpurnia

- Which plays of Shakespeare contain the words Brutus and Caesar, but not Calpurnia?
- Naive solution: linear scan through all text – "grepping"
- In this case, works OK (Shakespeare's Collected works has less than 1M words).
- But in the general case, with much larger text colletions, we need to index.
- Indexing is an offline operation that collects data about which words occur in a text, so that at search time you only have to access the precompiled index.

The term-document incidence matrix

Main idea: record for each document whether it contains each word out of all the different words Shakespeare used (about 32K).

	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
...						

Matrix element (t, d) is 1 if the play in column d contains the word in row t, 0 otherwise.

Query "Brutus AND Caesar AND NOT Calpunia"

We compute the results for our query as the bitwise AND between vectors for Brutus, Caesar and complement (Calpurnia):

	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
...						

Query "Brutus AND Caesar AND NOT Calpunia"

We compute the results for our query as the bitwise AND between vectors for Brutus, Caesar and complement (Calpurnia):

	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	1	0	1	1	1	1
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0
...						

Query "Brutus AND Caesar AND NOT Calpunia"

We compute the results for our query as the bitwise AND between vectors for Brutus, Caesar and complement (Calpurnia):

	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	1	0	1	1	1	1
Cleopatra	1	0	0	0	0	0
mercy	1	0	1	1	1	1
worser	1	0	1	1	1	0
AND	1	0	0	1	0	0

Bitwise AND returns two documents, "Antony and Cleopatra" and "Hamlet".

The results: two documents

Antony and Cleopatra, Act III, Scene ii

Agrippa [Aside to Dominitus Enobarbus]: Why, Enobarbus,
When Antony found Julius Caesar dead,
He cried almost to roaring, and he wept
When at Philippi he found Brutus slain.

Hamlet, Act III, Scene ii

Lord Polonius: I did enact Julius Caesar: I was killed i' the
Capitol; Brutus killed me.

Bigger collections

- Consider N=10^6 documents, each with about 1000 tokens
- 10^9 tokens at avg 6 Bytes per token $\Rightarrow$ 6GB
- Assume there are M=500,000 distinct terms in the collection
- Size of incidence matrix is then 500,000 $\times 10^6$
- Half a trillion 0s and 1s

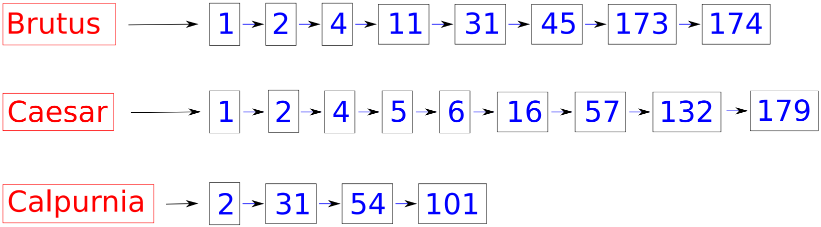
Can't build the Term-Document incidence matrix

- Observation: the term-document matrix is very sparse
- Contains no more than one billion 1s.
- Better representation: only represent the things that do occur
- Term-document matrix has other disadvantages, such as lack of support for more complex query operators (e.g., proximity search)
- We will move towards richer representations, beginning with the inverted index.

The inverted index

The inverted index consists of

- a dictionary of terms (also: lexicon, vocabulary)
- and a postings list for each term, i.e., a list that records which documents the term occurs in.

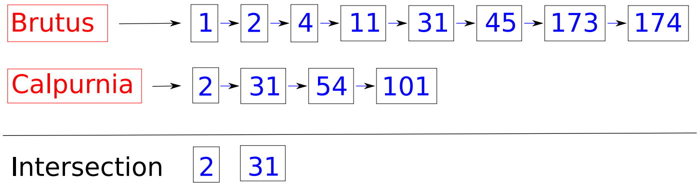


Processing Boolean Queries: conjunctive queries

Our Boolean Query

`Brutus AND Calpurnia`

Locate the postings lists of both query terms and intersect them.



Note: this only works if postings lists are sorted

Algorithm for intersection of two postings

```
INTERSECT (p1, p2)
answer ← <>
while p1 ≠ NIL and p2 ≠ NIL
do if docID(p1) = docID(p2)
    then ADD (answer, docID(p1))
        p1 ← next(p1)
        p2 ← next(p2)
    if docID(p1) < docID(p2)
        then p1← next(p1)
        else p2← next(p2)
 return answer
```

$$\begin{tikzcd}
\boxed{\text{Brutus}} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{11} \arrow[r] & \boxed{31} \arrow[r] & \boxed{45} \arrow[r] & \boxed{173} \arrow[r] & \boxed{174} \\
\boxed{\text{Calpurnia}} \arrow[r] & \boxed{2} \arrow[r] & \boxed{31} \arrow[r] & \boxed{54} \arrow[r] & \boxed{101}
\end{tikzcd}$$

Intersection $\boxed{2}$ $\boxed{31}$

Complexity of the Intersection Algorithm

- Bounded by worst-case length of postings lists
- Thus “officially” $O(N)$, with N the number of documents in the document collection
- But in practice much, much better than linear scanning, which is asymptotically also $O(N)$

Query Optimisation: conjunctive terms

Organise order in which the postings lists are accessed so that least work needs to be done

Brutus AND Caesar AND Calpurnia

Process terms in increasing document frequency: execute as

(Calpurnia AND Brutus) AND Caesar

$$\begin{tikzcd}
\boxed{\text{Brutus}}\ \boxed{8} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{11} \arrow[r] & \boxed{31} \arrow[r] & \boxed{45} \arrow[r] & \boxed{173} \arrow[r] & \boxed{174} \\
\boxed{\text{Caesar}}\ \boxed{9} \arrow[r] & \boxed{1} \arrow[r] & \boxed{2} \arrow[r] & \boxed{4} \arrow[r] & \boxed{5} \arrow[r] & \boxed{6} \arrow[r] & \boxed{16} \arrow[r] & \boxed{57} \arrow[r] & \boxed{132} \arrow[r] & \boxed{179} \\
\boxed{\text{Calpurnia}}\ \boxed{4} \arrow[r] & \boxed{2} \arrow[r] & \boxed{31} \arrow[r] & \boxed{54} \arrow[r] & \boxed{101}
\end{tikzcd}$$

Query Optimisation: disjunctive terms

(maddening OR crowd) AND (ignoble OR strife) AND (killed OR slain)

- Process the query in increasing order of the size of each disjunctive term
- Estimate this in turn (conservatively) by the sum of frequencies of its disjuncts

Practical Boolean Search

- Provided by large commercial information providers 1960s-1990s
- Complex query language; complex and long queries
- Extended Boolean retrieval models with additional operators – proximity operators
- Proximity operator: two terms must occur close together in a document (in terms of certain number of words, or within sentence or paragraph)
- Unordered results...

Example: Westlaw

- Largest commercial legal search service – 500K subscribers
- Boolean Search and ranked retrieval both offered
- Document ranking only wrt chronological order
- Expert queries are carefully defined and incrementally developed

Westlaw Queries/Information Needs

“trade secret” /s disclos! /s prevent /s employe!

Information need: Information on the legal theories involved in preventing the disclosure of trade secrets by employees formerly employed by a competing company.

disab! /p access! /s work-site work-place (employment /3 place)

Information need: Requirements for disabled people to be able to access a workplace.

host! /p (responsib! liab!) /p (intoxicat! drunk!) /p guest

Information need: Cases about a host’s responsibility for drunk guests.

Comments on WestLaw

- Proximity operators: /3= within 3 words, /s=within same sentence /p =within a paragraph
- Space is disjunction, not conjunction (This was standard in search pre-Google.)
- Long, precise queries: incrementally developed, unlike web search
- Why professional searchers like Boolean queries: **precision, transparency, control.**

Does Google use the Boolean Model?

On Google, the default interpretation of a query [w_1 w_2 ... w_n] is w_1 AND w_2 AND ... AND w_n

- Cases where you get hits which don't contain one of the $w-i$:
  - Page contains variant of w_i (morphology, misspelling, synonym)
  - long query (n is large)
  - Boolean expression generates very few hits
  - w_i was in the *anchor text*
- Google also *ranks* the result set
  - Simple Boolean Retrieval returns matching documents in no particular order.
  - Google (and most well-designed Boolean engines) rank hits according to some estimator of relevance

Reading

- Manning, Raghavan, Schütze: Introduction to Information Retrieval (MRS), chapter 1