Multimedia Information Retrieval and Technology:

1. Introduction

By : Fangyu Wu

Room: SD555

Tel. no. 88161780

Xi'an Jiaotong-Liverpool University

西交利物浦大学

Admin Messages

Office Location: SD555

Office hour: Wednesday 1:00-2:00pm

Attendence.

- Attendance should be recorded for all students.
- To monitor the classroom attendance of on-site international students as this is likely to be a condition of any international student study visa.

Xi'an Jiaotong-Liverpool University
西交利物浦大學

Sequence		Method	Assessment Type (EXAM or CW)	Duration	% of final mark	Resit (Y/N/S)[1]
001		Written Examination	EXAM	3 hour(s)	70	No
002	Lab Report 1	CW			15	No
003	Lab Report 2	CW			15	No

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Essential texts

Title	Author	ISBN/Publisher
INTRODUCTION TO INFORMATION RETRIEVAL	CHRISTOPHER D.MANNING, PRABHAKAR RAGHAVAN	CAMBRIDGE
FUNDAMENTALS OF MULTIMEDIA	ZE-NIAN LI AND MARK S. DREW	PRENTICE-HAL

Xi'an Jiaotong-Liverpool University
西交利物浦大学

Other Good IR Books

1. *Managing Gigabytes*, by I. Witten, A. Moffat, and T. Bell.
2. *Information Retrieval: Algorithms and Heuristics* by D. Grossman and O. Frieder.
3. *Modern Information Retrieval*, by R. Baeza-Yates and B. Ribeiro-Neto.
4. *Foundations of Statistical Natural Language Processing*, by C. Manning and H. Schütze.
5. *Search Engines: Information Retrieval in Practice*, by Bruce Croft, Donald Metzler and Trevor Strohman.
6. *Information Retrieval: Implementing and Evaluating Search Engines*, by Stefan Buettcher, Charles L. A. Clarke and Gordon V. Cormack.

Course Structure-1

1) Text Information Retrieval

http://www-nlp.stanford.edu/IR-book/

PS: You can download and print chapters here for the book "INTRODUCTION TO INFORMATION RETRIEVAL"

- How to do efficient (fast, compact) text indexing
- Retrieval models: Boolean, vector-space, probabilistic, and machine learning models
- Evaluation
- Document clustering and classification

Xi'an Jiaotong-Liverpool University 西交利物浦大学

Course Structure-2

2) Image, Audio, Video, and their Retrieval

- Audio signal
- Image data representations
- Compression and retrieval

Agenda

- Introducing Information Retrieval
- A term-document matrix
- The inverted index data structure

Information Retrieval

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

Basic assumptions of Information Retrieval

Information need: the topic about which the user desires to know more.

Query: what the user conveys to the computer in an attempt to communicate the information need.

10

Basic assumptions of Information Retrieval

Collection: A set of documents.

Assume it is a static collection at the moment.

Goal: Retrieve documents with information that is relevant to the user's information need and helps the user complete a task.

Information Retrieval

- Web search
  - Search over billions of documents stored on millions of computers.
- Enterprise, institutional and domain-specific search
  - A database of patents, a corporations internal documents.

Information Retrieval

- Personal information retrieval
  - Some consumer operating systems have integrated information retrieval.
  - Email programs usually not only provide search but also text classification
  - Issues: handling the broad range of document types on a typical personal computer, and making the search system lightweight in terms of startup, processing and disk space usage.

Information Retrieval

"Unstructured data":

Data that does not have clear, semantically overt, easy-for-a-computer structure.

In modern parlance, the word "search" has tended to replace "information retrieval".

IR is fast becoming the dominant form of information access, overtaking traditional database style searching.

Structured vs unstructured data

Structured data tends to refer to information in "tables"

Employee	Manager	Salary
Smith	Jones	50000
Chang	Smith	60000
Ivy	Smith	50000

Typically allows numerical range and exact match (for text) queries, e.g.,

Salary < 60000 AND Manager = Smith.

Xi'an Jiaotong-Liverpool University 西交利物浦大学

15

Unstructured data

- Typically refers to free text
- Allows
  - Keyword queries including operators
    - And , or, but not…/S
  - More sophisticated "concept" queries e.g.,
    - find all web pages dealing with *red apple*
- Classic model for searching text documents

Term-document incidence matrices

An example

- Suppose you want to know which plays of Shakespeare contain the words ***Brutus*** *AND* ***Caesar*** but *NOT* ***Calpurnia***?
- One could `grep` all of Shakespeare's plays for ***Brutus*** and ***Caesar,*** then strip out lines containing ***Calpurnia***?
- Why is that not the answer?
  - Slow
  - Other operations (e.g., find the word ***Romans*** near ***countrymen***) not feasible
  - To allow ranked retrieval.

Xi'an Jiaotong-Liverpool University
西交利物浦大学

An example

The way to avoid linearly scanning the texts for each query is to index the documents in advance.

We record for each document, whether it contains each word out of all the words in the collection.

The result is a binary term-document incidence matrix.

Term-document incidence matrices

	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

1 if play contains word, 0 otherwise

Brutus *AND* ***Caesar*** *BUT NOT* ***Calpurnia***

Incidence vectors

So we have a 0/1 vector for each term.

To answer query: take the vectors for ***Brutus*** AND ***Caesar*** AND NOT ***Calpurnia*** (complemented) ➔ bitwise *AND*.

110100 *AND*

110111 *AND*

101111 =

100100

	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

Answers to query

Antony and Cleopatra, Act III, Scene ii

Agrippa [Aside to DOMITIUS 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.

Boonlean retrieval model

The boolean retrieval model is a model for IR in which we can pose any query which is in the form of a Boolean expression of terms, that is, the terms are combined with the operators AND, OR, and NOT.

The model views each document as just a set of words.

Bigger collections

10^6

Consider N = 1 million documents, each with about 1000 words. 10^3

Documents: we mean whatever units we have decided to build a retrieval system. They might be individual memos or chapters of a book.

The group of documents over which we perform retrieval is referred to as ***the collection***

$N = 10^6$ documents

$N_W = 10^3$ words/doc

$N_M = 6$ bytes/word

$N_{size} = 10^6 \times 10^3 \times 6$

$= 6 \times 10^9$ bytes

$= \dfrac{6 \times 10^9}{(10^3)^3}$ GB

Assume average 6 bytes/word including spaces and punctuation.

$= 6$ GB

The size of this document collection: 6GB.

Say there are *M* = 500K *distinct* terms among these.

docs

terms

10^6 documents

500K terms

matrix size $= 5 \times 10^5 \times 10^6$

$= 5 \times 10^{11}$

(half a trillion)

Assume average 6 bytes/word including spaces and punctuation.

The size of this document collection: 6GB.

Say there are *M* = 500K *distinct* terms among these.

The term-document matrix will be a 500K*1M matrix with half a trillion 0's and 1's.

Can’t build the matrix

Too many to fit in a computer’s memory!

But it has no more than one billion 1’s.

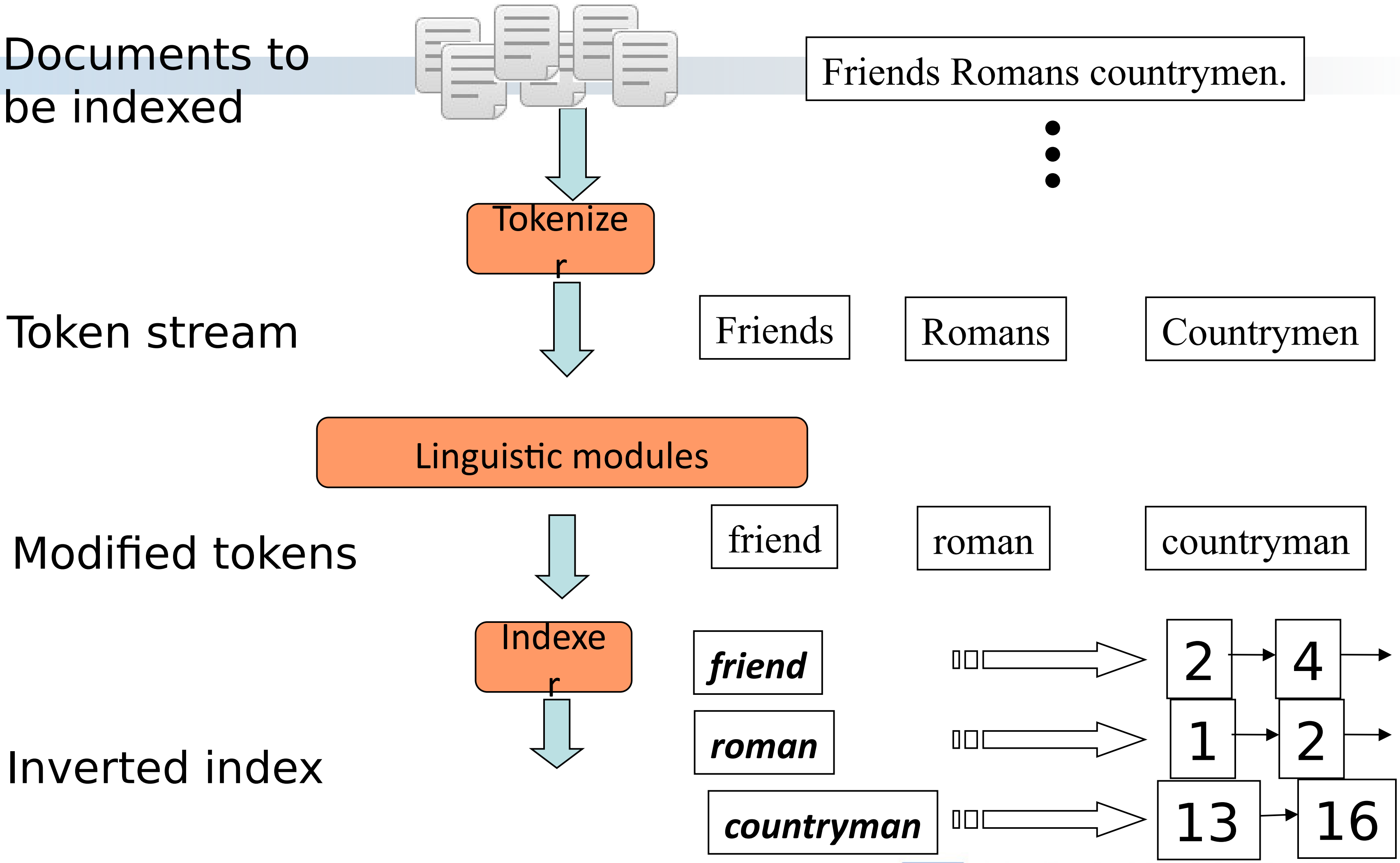
Why?

the matrix is extremely sparse, 99.8% of the cells are 0.

What’s a better representation?

The Inverted Index

Inverted index construction



Inverted index

Inverted index: an index always maps back from terms to the parts of a document where they occur.

The two parts of an inverted index: Dictionary and Postings. The dictionary is commonly kept in memory, with pointers to each posting list, which is stored on disk.

Inverted index

For each term *t*, we must store a list of all documents that contain *t*. Identify each doc by a **docID**, a unique document serial number

Can we used fixed-size arrays for this?

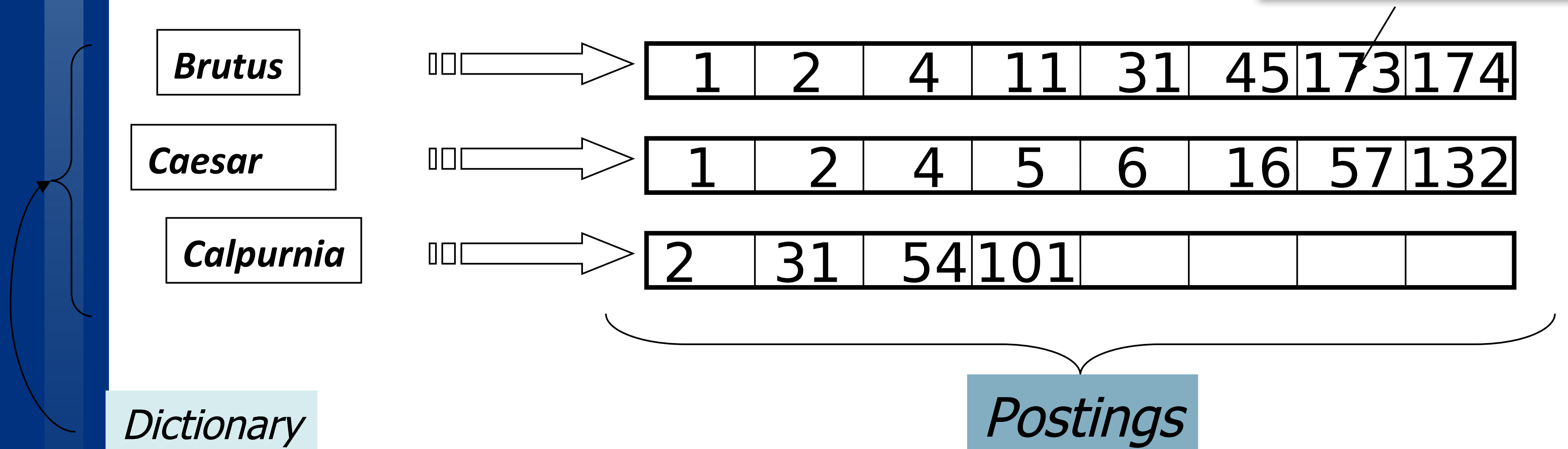
Brutus →	1	2	4	11	31	45	173	174
Caesar →	1	2	4	5	6	16	57	132
Calpurnia →	2	31	54	101				

What happens if the word ***Caesar*** is added to document 14?

Inverted index

We need variable-size postings lists

- On disk, a continuous run of postings is normal and best
- In memory, can use linked lists or variable length arrays
  - Some tradeoffs in size/ease of insertion



Sorted by docID (more later on why).

Basic concepts

Dictionary: sometimes referred to as a vocabulary or lexicon.

Posting: each item in the list records documents that the term appeared in. Then the list is called a postings list or inverted list. And all the postings lists taken together are referred to as the postings.

The heuristic

The input to indexing is a list of normalized tokens for each document, which we can equally think of as a list of pairs of term and docID.

1. Sorting the list so that the terms are alphabetical;
2. Multiple occurrences of the same term from the same document are merged.

Indexer steps: Token sequence

Sequence of (Modified token, Document ID) pairs.

Doc 1

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

Doc 2

So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious

Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Xi'an Jiaoton
西交利

Indexer steps: Sort

Sort by terms
And then docID

Core indexing step

Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

Indexer steps: Dictionary & Postings

- Multiple term entries in a single document are merged.
- Split into Dictionary and Postings
- Doc. frequency information is added.

Term	docID
ambitious	2
be	2
brutus	1
brutus	2
capitol	1
caesar	1
caesar	2
caesar	2
did	1
enact	1
hath	1
I	1
I	1
i'	1
it	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	1
was	2
with	2

term	doc. freq.	→	postings lists
ambitious	1	→	2
be	1	→	2
brutus	2	→	1 → 2
capitol	1	→	1
caesar	2	→	1 → 2
did	1	→	1
enact	1	→	1
hath	1	→	2
i	1	→	1
i'	1	→	1
it	1	→	2
julius	1	→	1
killed	1	→	1
let	1	→	2
me	1	→	1
noble	1	→	2
so	1	→	2
the	2	→	1 → 2
told	1	→	2
you	1	→	2
was	2	→	1 → 2
with	1	→	2

Why frequency?
Will discuss later.

The postings are sorted by docID, this provides the basis for efficient query processing.

The inverted index structure is essentially without rival as the most efficient structure for supporting ad hoc text search.

Exercise

Consider these documents:

Doc 1 breakthrough drug for schizophrenia

Doc 2 new schizophrenia drug

Doc 3 new approach for treatment of schizophrenia

Doc 4 new hopes for schizophrenia patients

Draw the term-document incidence matrix for this document collection.

Draw the inverted index representation for this collection.

	d1	d2	d3	d4
Approach	0	0	1	0
breakthrough	1	0	0	0
drug	1	1	0	0
for	1	0	1	1
hopes	0	0	0	1
new	0	1	1	1
of	0	0	1	0
patients	0	0	0	1
schizophrenia	1	1	1	1
treatment	0	0	1	0

Inverted Index:

Approach -> 3

breakthrough ->1

drug ->1->2

for ->1->3->4

hopes ->4

new -.>2->3->4

of ->3

patients ->4

schizophrenia ->1->2->3->4

treatment >3