

# Introduction to Information Retrieval

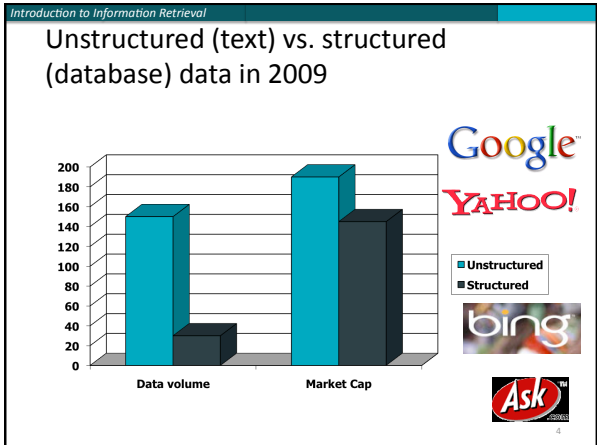
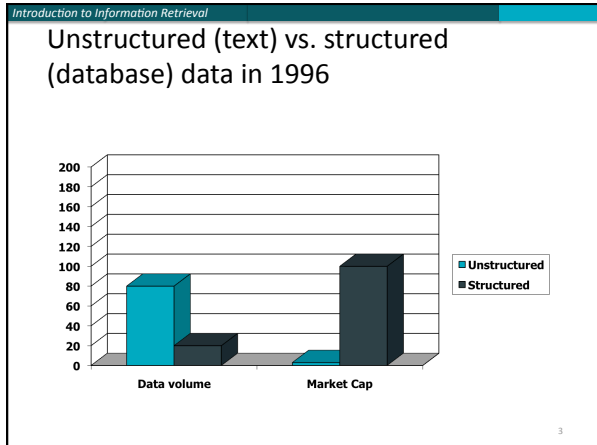
CS276  
Information Retrieval and Web Search  
Pandu Nayak and Prabhakar Raghavan  
Lecture 1: Boolean retrieval

Introduction to Information Retrieval

## Information Retrieval

- 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).

2



Introduction to Information Retrieval

Sec. 1.1

### Unstructured data in 1680

- 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 (for large corpora)
  - NOT Calpurnia** is non-trivial
  - Other operations (e.g., find the word **Romans** near **countrymen**) not feasible
  - Ranked retrieval (best documents to return)
    - Later lectures

5

Introduction to Information Retrieval

Sec. 1.1

### Term-document incidence

	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

**Brutus AND Caesar BUT NOT Calpurnia**

1 if play contains word, 0 otherwise

Introduction to Information Retrieval Sec. 1.1

## Incidence vectors


- So we have a 0/1 vector for each term.
- To answer query: take the vectors for **Brutus**, **Caesar** and **Calpurnia** (complemented) → bitwise AND.
- 110100 AND 110111 AND 101111 = 100100.

7

Introduction to Information Retrieval Sec. 1.1

## 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.



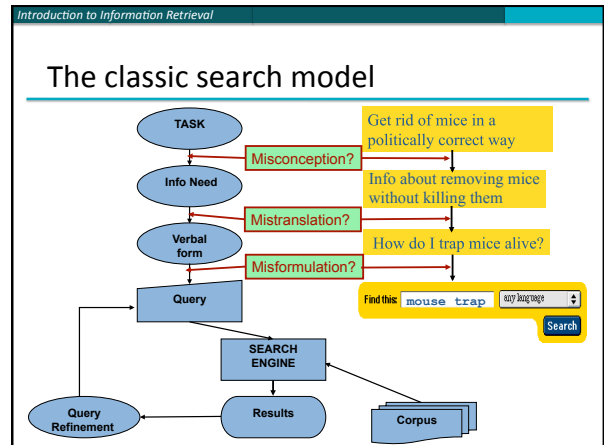
8

Introduction to Information Retrieval Sec. 1.1

## Basic assumptions of Information Retrieval

- Collection**: Fixed set of documents
- Goal**: Retrieve documents with information that is relevant to the user's **information need** and helps the user complete a **task**

9



Introduction to Information Retrieval Sec. 1.1

## How good are the retrieved docs?

- Precision**: Fraction of retrieved docs that are relevant to user's information need
- Recall**: Fraction of relevant docs in collection that are retrieved
- More precise definitions and measurements to follow in later lectures

11

Introduction to Information Retrieval Sec. 1.1

## Bigger collections

- Consider  $N = 1$  million documents, each with about 1000 words.
- Avg 6 bytes/word including spaces/punctuation
  - 6GB of data in the documents.
- Say there are  $M = 500K$  *distinct* terms among these.

12

Introduction to Information Retrieval Sec. 1.1

## Can't build the matrix

- 500K x 1M matrix has half-a-trillion 0's and 1's.
- But it has no more than one billion 1's. Why?
  - matrix is extremely sparse.
- What's a better representation?
  - We only record the 1 positions.

13

Introduction to Information Retrieval Sec. 1.2

## Inverted index

- For each term  $t$ , we must store a list of all documents that contain  $t$ .
  - Identify each by a **docID**, a document serial number
- Can we use fixed-size arrays for this?

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

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

14

Introduction to Information Retrieval Sec. 1.2

## 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

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

Posting

Dictionary

Postings

Sorted by docID (more later on why).

Introduction to Information Retrieval Sec. 1.2

## Inverted index construction

Introduction to Information Retrieval Sec. 1.2

## 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

Introduction to Information Retrieval Sec. 1.2

## Indexer steps: Sort

- Sort by terms
  - And then docID

Core indexing step

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

Introduction to Information Retrieval Sec. 1.2

## 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
claudius	1
enact	1
enact	2
hath	1
i	1
i	1
f	1
f	1
e	2
julius	1
killed	1
killed	1
let	2
me	1
noble	2
so	2
the	1
the	2
told	2
you	2
was	2
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
enact	1	→ 2
hath	1	→ 1
i	1	→ 1
f	1	→ 1
f	1	→ 2
e	2	→ 1
julius	1	→ 1
killed	1	→ 1
killed	1	→ 1
let	2	→ 1
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.

Introduction to Information Retrieval Sec. 1.2

## Where do we pay in storage?

Terms and counts

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
enact	1	→ 2
hath	1	→ 1
i	1	→ 1
f	1	→ 1
f	1	→ 2
e	2	→ 1
julius	1	→ 1
killed	1	→ 1
killed	1	→ 1
let	2	→ 1
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

Lists of docIDs

Pointers

Later in the course:  
• How do we index efficiently?  
• How much storage do we need?

Introduction to Information Retrieval Sec. 1.3

## The index we just built

- How do we process a query?
  - Later - what kinds of queries can we process?

Today's focus

Introduction to Information Retrieval Sec. 1.3

## Query processing: AND

- Consider processing the query:
  - Brutus AND Caesar**
    - Locate **Brutus** in the Dictionary;
      - Retrieve its postings.
    - Locate **Caesar** in the Dictionary;
      - Retrieve its postings.
    - “Merge” the two postings:

Introduction to Information Retrieval Sec. 1.3

## The merge

- Walk through the two postings simultaneously, in time linear in the total number of postings entries

If list lengths are  $x$  and  $y$ , merge takes  $O(x+y)$  operations.  
Crucial: postings sorted by docID.

Introduction to Information Retrieval Sec. 1.3

## Intersecting two postings lists (a “merge” algorithm)

```

INTERSECT( $p_1, p_2$ )
1  answer ←  $\langle \rangle$ 
2  while  $p_1 \neq \text{NIL}$  and  $p_2 \neq \text{NIL}$ 
3  do if  $\text{docID}(p_1) = \text{docID}(p_2)$ 
4  then ADD(answer,  $\text{docID}(p_1)$ )
5          $p_1 \leftarrow \text{next}(p_1)$ 
6          $p_2 \leftarrow \text{next}(p_2)$ 
7  else if  $\text{docID}(p_1) < \text{docID}(p_2)$ 
8  then  $p_1 \leftarrow \text{next}(p_1)$ 
9  else  $p_2 \leftarrow \text{next}(p_2)$ 
10 return answer
    
```



Introduction to Information Retrieval Sec. 1.3

## Query optimization example

- Process in order of increasing freq:
  - start with smallest set, then keep cutting further.

This is why we kept document freq. in dictionary

<b>Brutus</b>	⇒	2	4	8	16	32	64	128	
<b>Caesar</b>	⇒	1	2	3	5	8	16	21	34
<b>Calpurnia</b>	⇒	13	16						

Execute the query as **(Calpurnia AND Brutus) AND Caesar**.

31

Introduction to Information Retrieval Sec. 1.3

## More general optimization

- e.g., **(madding OR crowd) AND (ignoble OR strife)**
- Get doc. freq.'s for all terms.
- Estimate the size of each OR by the sum of its doc. freq.'s (conservative).
- Process in increasing order of OR sizes.

32

Introduction to Information Retrieval

## Exercise

- Recommend a query processing order for

	<b>Term</b>	<b>Freq</b>
<i>(tangerine OR trees) AND</i>	eyes	213312
<i>(marmalade OR skies) AND</i>	kaleidoscope	87009
<i>(kaleidoscope OR eyes)</i>	marmalade	107913
	skies	271658
	tangerine	46653
	trees	316812

33

Introduction to Information Retrieval

## Query processing exercises

- Exercise:** If the query is **friends AND romans AND (NOT countrymen)**, how could we use the freq of **countrymen**?
- Exercise:** Extend the merge to an arbitrary Boolean query. Can we always guarantee execution in time linear in the total postings size?
- Hint:** Begin with the case of a Boolean *formula* query where each term appears only once in the query.

34

Introduction to Information Retrieval

## Exercise

- Try the search feature at <http://www.rhymezone.com/shakespeare/>
- Write down five search features you think it could do better

35

Introduction to Information Retrieval

## What's ahead in IR? Beyond term search

- What about phrases?
  - Stanford University**
- Proximity: Find **Gates NEAR Microsoft**.
  - Need index to capture position information in docs.
- Zones in documents: Find documents with *(author = Ullman) AND* (text contains **automata**).

36

## Evidence accumulation

- 1 vs. 0 occurrence of a search term
  - 2 vs. 1 occurrence
  - 3 vs. 2 occurrences, etc.
  - Usually more seems better
- Need term frequency information in docs

37

## Ranking search results

- Boolean queries give inclusion or exclusion of docs.
- Often we want to rank/group results
  - Need to measure proximity from query to each doc.
  - Need to decide whether docs presented to user are singletons, or a group of docs covering various aspects of the query.

38

## IR vs. databases:

### 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.*

39

## Unstructured data

- Typically refers to free-form text
- Allows
  - Keyword queries including operators
  - More sophisticated “concept” queries, e.g.,
    - find all web pages dealing with *drug abuse*
- Classic model for searching text documents

40

## Semi-structured data

- In fact almost no data is “unstructured”
- E.g., this slide has distinctly identified zones such as the *Title* and *Bullets*
- Facilitates “semi-structured” search such as
  - *Title* contains data AND *Bullets* contain search

... to say nothing of linguistic structure

41

## More sophisticated semi-structured search

- *Title* is about Object Oriented Programming AND *Author* something like stro\*rup
- where \* is the wild-card operator
- Issues:
  - how do you process “about”?
  - how do you rank results?
- The focus of XML search (*IIR* chapter 10)

42

## Clustering, classification and ranking

- **Clustering:** Given a set of docs, group them into clusters based on their contents.
- **Classification:** Given a set of topics, plus a new doc  $D$ , decide which topic(s)  $D$  belongs to.
- **Ranking:** Can we learn how to best order a set of documents, e.g., a set of search results

43

## The web and its challenges

- Unusual and diverse documents
- Unusual and diverse users, queries, information needs
- Beyond terms, exploit ideas from social networks
  - link analysis, clickstreams ...
- How do search engines work?  
And how can we make them better?

44

## More sophisticated *information* retrieval

- Cross-language information retrieval
- Question answering
- Summarization
- Text mining
- ...

45

## Course details

- Course URL: [cs276.stanford.edu](http://cs276.stanford.edu)
  - [a.k.a., <http://www.stanford.edu/class/cs276/> ]
- Work/Grading:
 

▪ Problem sets (2)	20%
▪ Practical exercises (2)	10% + 20% = 30%
▪ Midterm	20%
▪ Final	30%
- Textbook:
  - *Introduction to Information Retrieval*
    - In bookstore and online (<http://informationretrieval.org/>)
    - We're happy to get comments/corrections/feedback on it!

46

## Course staff

- **Professor:** [Pandu Nayak](mailto:nayak@cs.stanford.edu)  
[nayak@cs.stanford.edu](mailto:nayak@cs.stanford.edu)
- **Professor:** [Prabhakar Raghavan](mailto:pragh@cs.stanford.edu)  
[pragh@cs.stanford.edu](mailto:pragh@cs.stanford.edu)
- **TAs:** Sonali Aggarwal, Sandeep Sripada,  
Valentin Spitkovsky
- *In general, don't use the above addresses, but:*
  - Newsgroup: [su.class.cs276](mailto:su.class.cs276) [preferred]
  - [cs276-spr1011-staff@lists.stanford.edu](mailto:cs276-spr1011-staff@lists.stanford.edu)

47

## Resources for today's lecture

- *Introduction to Information Retrieval*, chapter 1
- Shakespeare:
  - <http://www.rhymezone.com/shakespeare/>
  - Try the neat browse by keyword sequence feature!
- *Managing Gigabytes*, chapter 3.2
- *Modern Information Retrieval*, chapter 8.2

Any questions?

48