CSCI 5417
Information Retrieval Systems

Lecture 1

8/23/2011

Introduction

1

What is Information Retrieval?

Information retrieval is the science of searching for information in documents, searching for documents themselves, searching for metadata which describe documents, or searching within databases, whether relational stand-alone databases or hypertextually-networked databases such as the World Wide Web.

Wikipedia

The study of methods and structures used to represent and access information.

Witten et al.

The IR definition can be found in this book.

Salton

IR deals with the representation, storage, organization of, and access to information items.

Salton

Information retrieval is the term conventionally, though somewhat inaccurately, applied to the type of activity discussed in this volume.

van Rijsbergen

2

Manning et al.

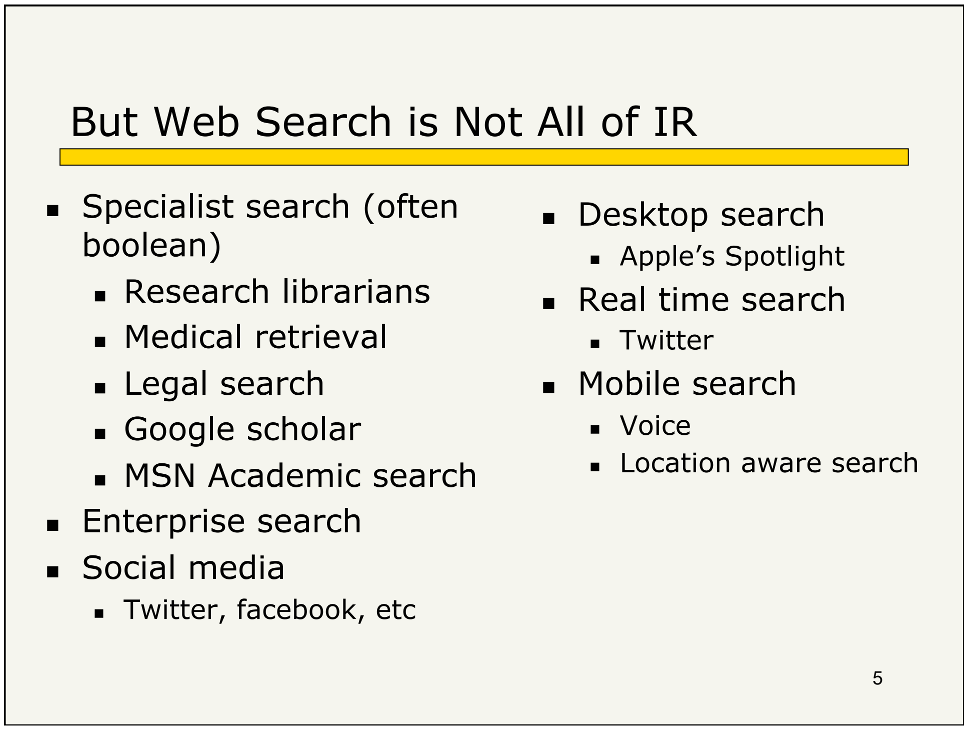
- Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

3

How about: *What you do when you use Google*

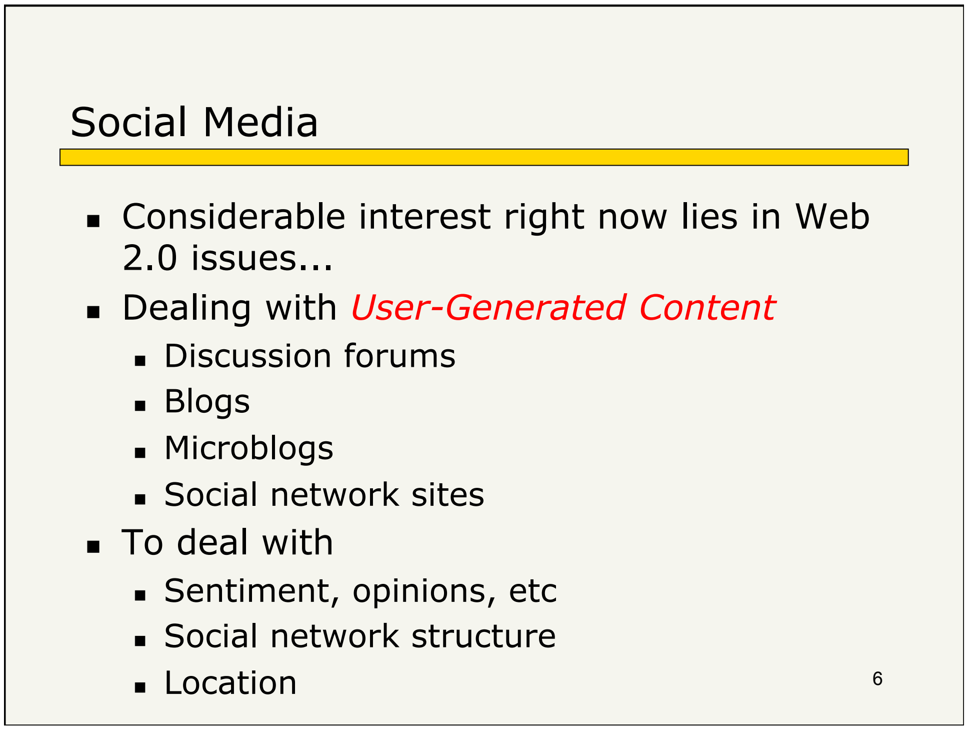
- *Ad hoc retrieval* is the core task that modern IR systems address
  - One-shot information seeking attempts by ignorant users
    - Ignorant about the structure and content of the collection
    - Ignorant about how the system works
    - Ignorant about how to formulate queries
  - Typically textual documents, but video and audio are becoming more prevalent
  - Collections are heterogeneous in nature

4



But Web Search is Not All of IR

- Specialist search (often boolean)
  - Research librarians
  - Medical retrieval
  - Legal search
  - Google scholar
  - MSN Academic search
- Enterprise search
- Social media
  - Twitter, facebook, etc
- Desktop search
  - Apple’s Spotlight
- Real time search
  - Twitter
- Mobile search
  - Voice
  - Location aware search



Social Media

- Considerable interest right now lies in Web 2.0 issues…
- Dealing with *User-Generated Content*
  - Discussion forums
  - Blogs
  - Microblogs
  - Social network sites
- To deal with
  - Sentiment, opinions, etc
  - Social network structure
  - Location

Web

7

Course Plan

- Cover the basics of IR technology in the first part of the course
  - The book provides the bulk of this material
- Investigate newer topics in the latter part
- Use discussions of real companies throughout the semester
- Project presentations and discussions for the last section of the class.

- I expect informed participation.

8

Course Plan

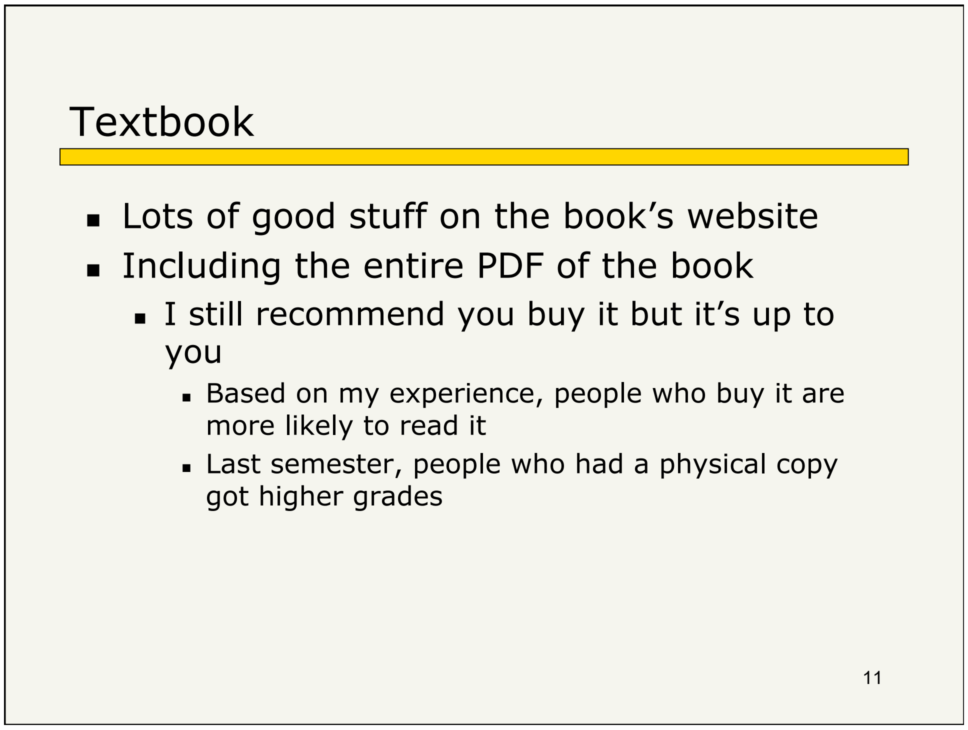
- Core technology areas
  - Indexing and ranked retrieval
    - Basic vector space model
    - Probabilistic models
    - Supervised ML ranking methods
  - Document classification
    - Sometimes called routing or filtering
    - Supervised ML approaches
  - Document clustering
    - Unsupervised and semi-supervised ML approaches

Administrivia

- Work/Grading

Programming assignments	30%
Quizzes	30%
Project	30%
Participation	10%

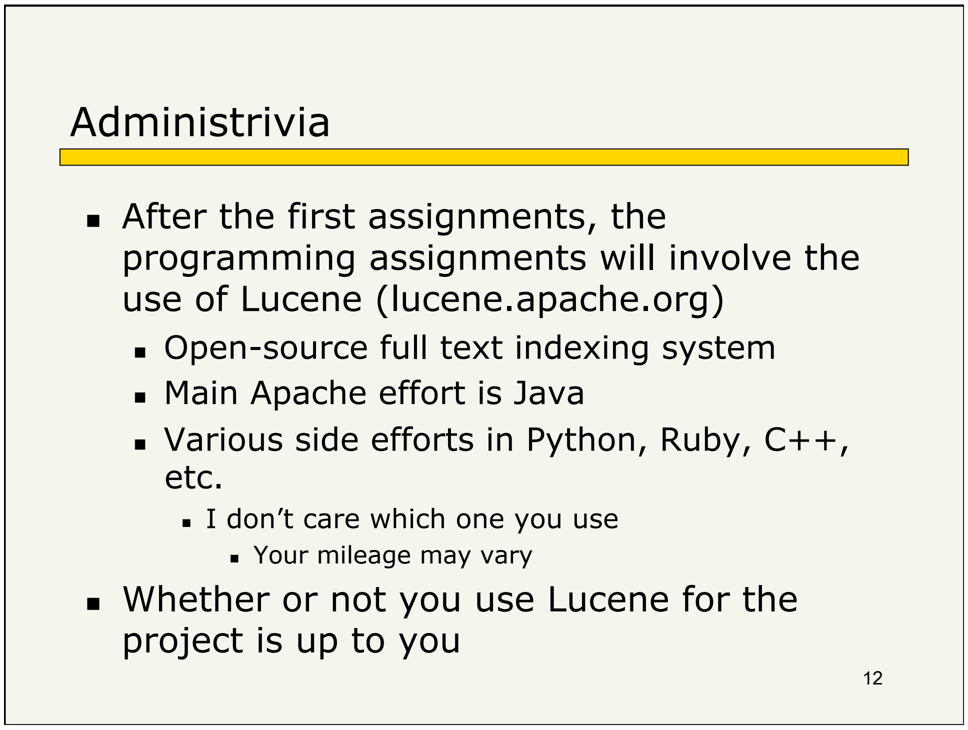
- Textbook
  - *Introduction to Information Retrieval --- Manning, Raghavan and Schütze*



Textbook

- Lots of good stuff on the book's website
- Including the entire PDF of the book
  - I still recommend you buy it but it's up to you
    - Based on my experience, people who buy it are more likely to read it
    - Last semester, people who had a physical copy got higher grades

11



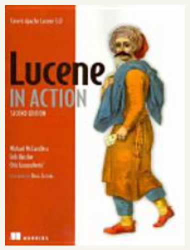
Administrivia

- After the first assignments, the programming assignments will involve the use of Lucene (lucene.apache.org)
  - Open-source full text indexing system
  - Main Apache effort is Java
  - Various side efforts in Python, Ruby, C++, etc.
    - I don't care which one you use
      - Your mileage may vary
- Whether or not you use Lucene for the project is up to you

12

Lucene Documentation

- Lucene has all the usual Java-doc style information associated with it. See the main Lucene page.
- The main reference text associated with it is "Lucene in Action" 2ed.
  - See the publisher page for more info. Chapter 1 is free.



13

CAETE

- Remote students have a 1 week offset for assignments and quizzes
  - With warning I'm flexible on the assignments
  - Less so for the quizzes
- For quizzes you need to have an EO or come to class for the quiz
- The #1 problem that CAETE students run into is falling behind on the lectures
  - For what its worth, that's also the #1 problem for local students who "attend" but are not prepared for class.

14

Web/Email

- Slides will be available as ppt and pdf shortly after each class
- You will be able to view the videos on the web. You should use this for review and for situations where you can't come to class (like the flu), not as a reason to skip class
- My roster has your colorado.edu email addresses. If you read your mail elsewhere then you need to set up a forward.

15

Piazza

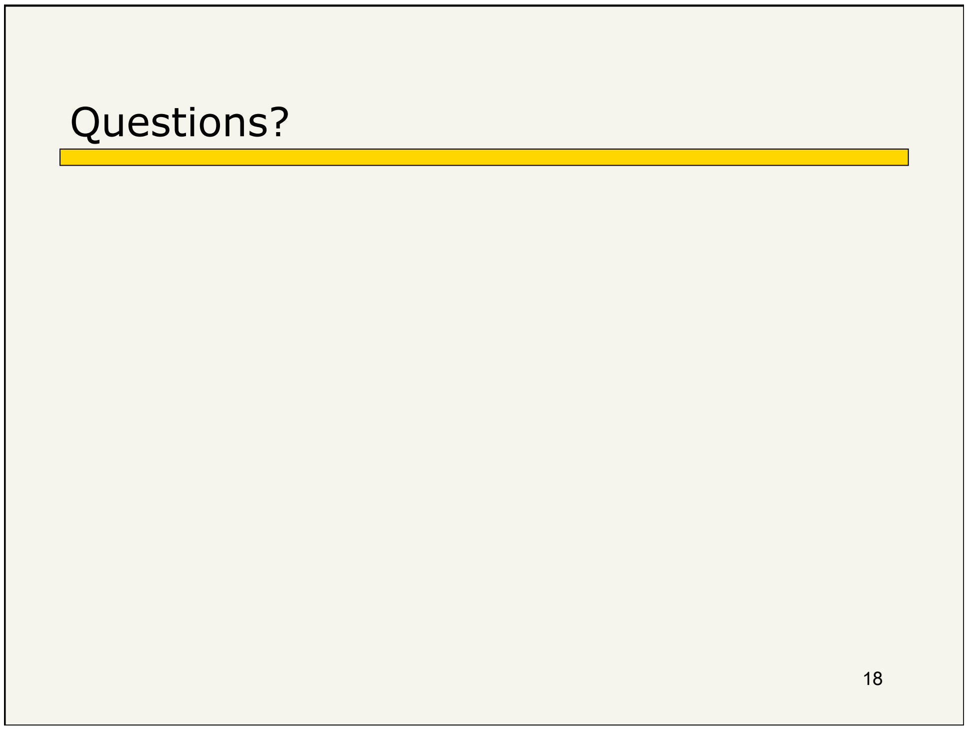
- We'll be using a new website as a resource for Q/A about the class.
  - Topics
  - Assignments
  - Quiz reviews etc.
- Go to Piazza and register for this class
  - At least some part of "participation" can be taken care of through your use of Piazza

16

Administrivia

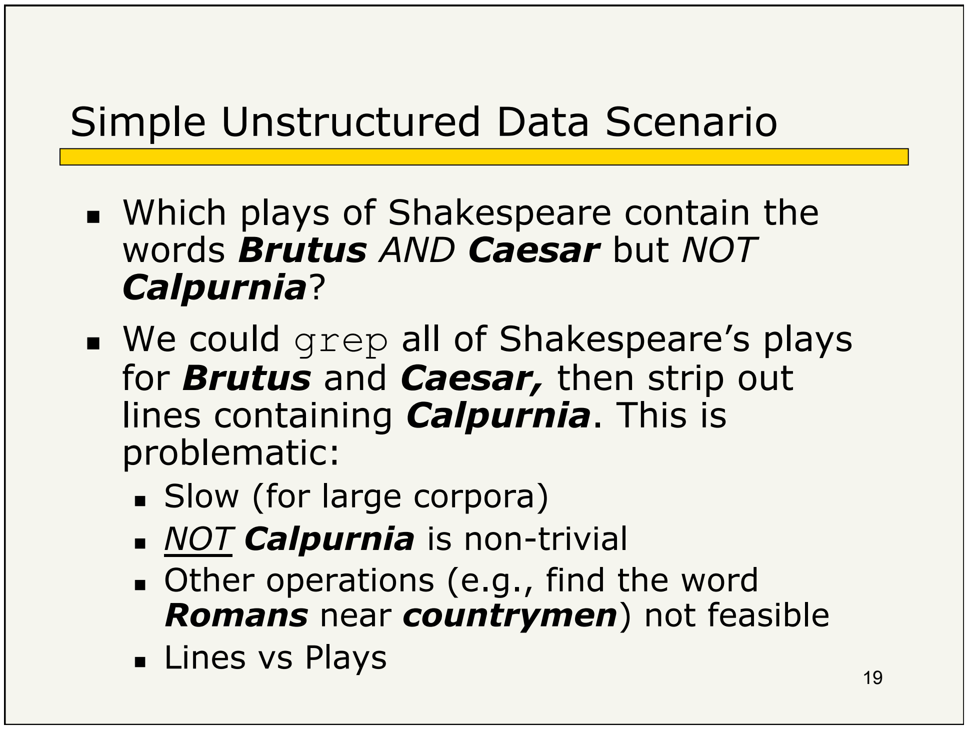
- **Professor: Jim Martin**
  - James.martin@colorado.edu
  - ECOT 726
  - Office hours TBA
  - www.cs.colorado.edu/~martin/csci5417/

17



Questions?

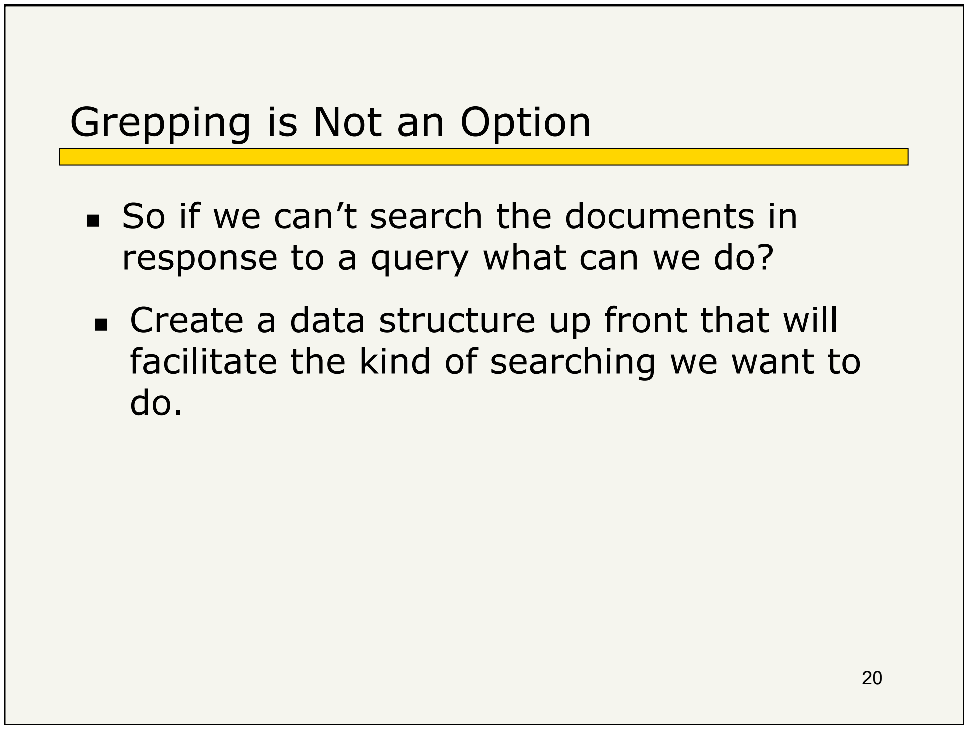
18



Simple Unstructured Data Scenario

- Which plays of Shakespeare contain the words ***Brutus*** *AND* ***Caesar*** but *NOT* ***Calpurnia***?
- We could `grep` all of Shakespeare's plays for ***Brutus*** and ***Caesar,*** then strip out lines containing ***Calpurnia***. This is problematic:
  - Slow (for large corpora)
  - *NOT* ***Calpurnia*** is non-trivial
  - Other operations (e.g., find the word ***Romans*** near ***countrymen***) not feasible
  - Lines vs Plays

19



Grepping is Not an Option

- So if we can't search the documents in response to a query what can we do?
- Create a data structure up front that will facilitate the kind of searching we want to do.

20

Term-Document Matrix

	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

Incidence Vectors

- So we have a 0/1 vector for each term
  - Length of the term vector = number of plays
- To answer our query: take the vectors for ***Brutus, Caesar*** and ***Calpurnia*** (complemented) and then do a bitwise *AND*.
- 110100 *AND* 110111 *AND* 101111 = 100100
  - That is, plays 1 and 4
  - "Antony and Cleopatra" and "Hamlet"

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.

23

Bigger Collections

- Consider *N* = 1M documents, each with about 1K terms.
- Avg 6 bytes/term including spaces and punctuation
  - 6GB of data just for the documents.
- Assume there are *m* = 500K *distinct* terms among these.
  - Types vs. Tokens

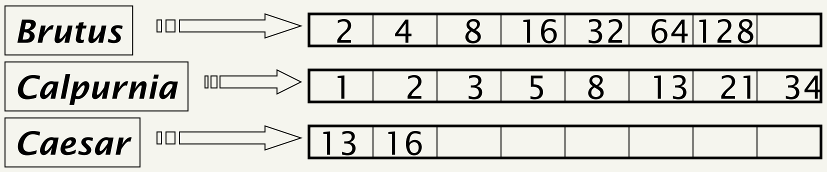
24

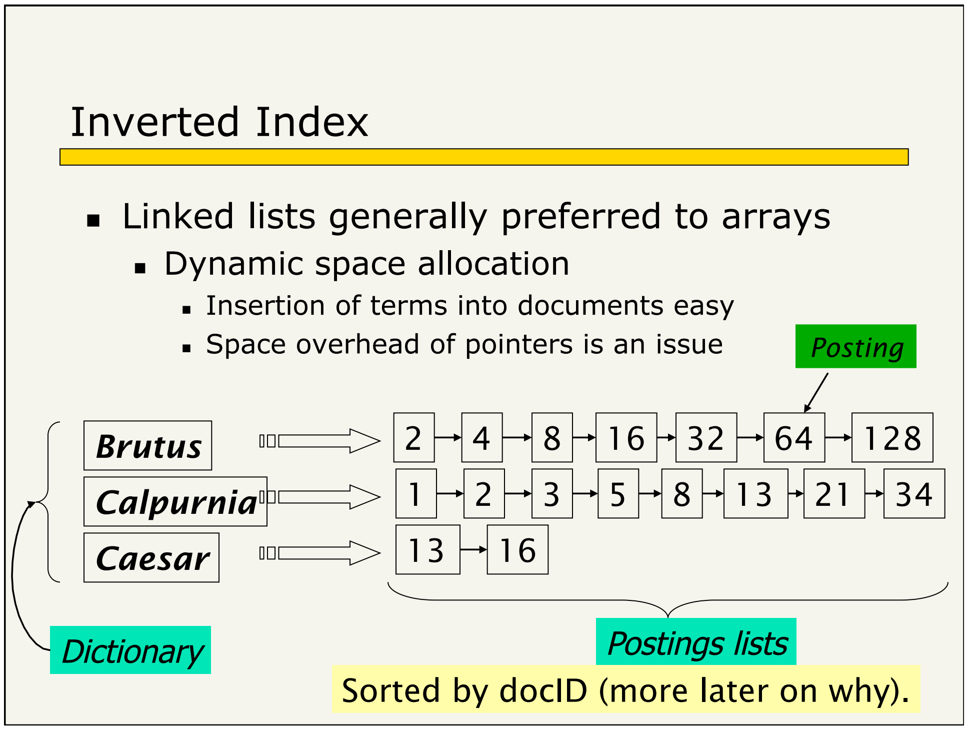
The Matrix

- 500K x 1M matrix has 1/2 trillion entries
- But it has no more than one billion 1's ← Why?
    - Matrix is extremely sparse.
    - What's the minimum number of 1's in such an index?
- What's a better representation?
    - Forget the 0's. Only record the 1's.

Inverted Index

- For each term *T*, we store a list of all documents that contain *T*.

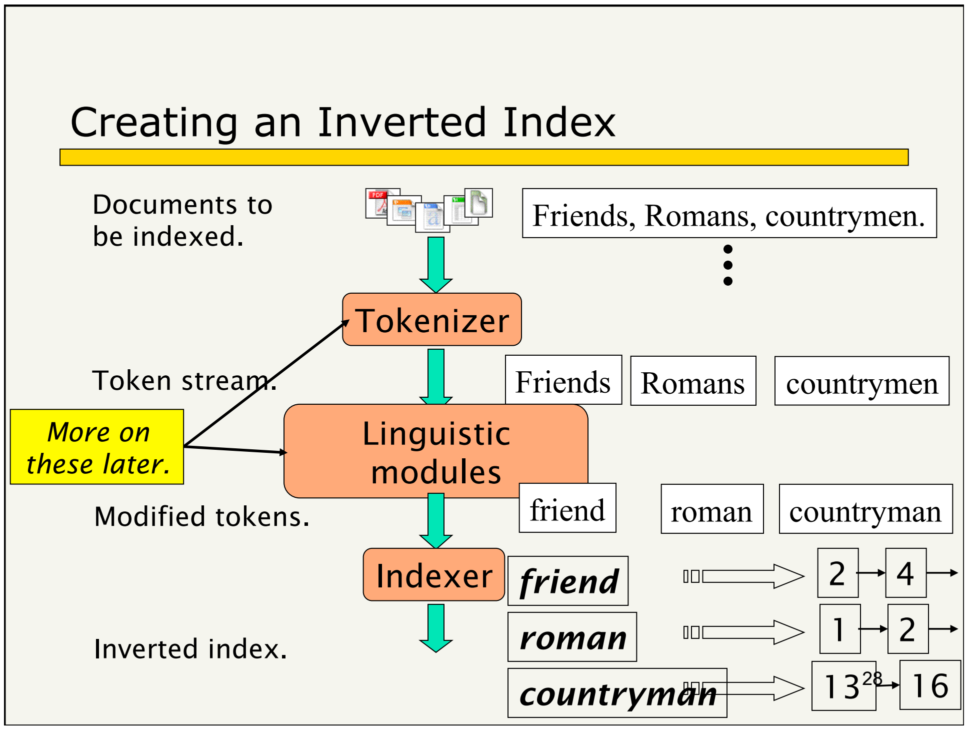




Inverted Index

- Linked lists generally preferred to arrays
  - Dynamic space allocation
    - Insertion of terms into documents easy
    - Space overhead of pointers is an issue

Sorted by docID (more later on why).



Creating an Inverted Index

Indexer steps: Token sequence

- First generate a sequence of <token, Document-ID> pairs from the stream of documents being indexed.

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

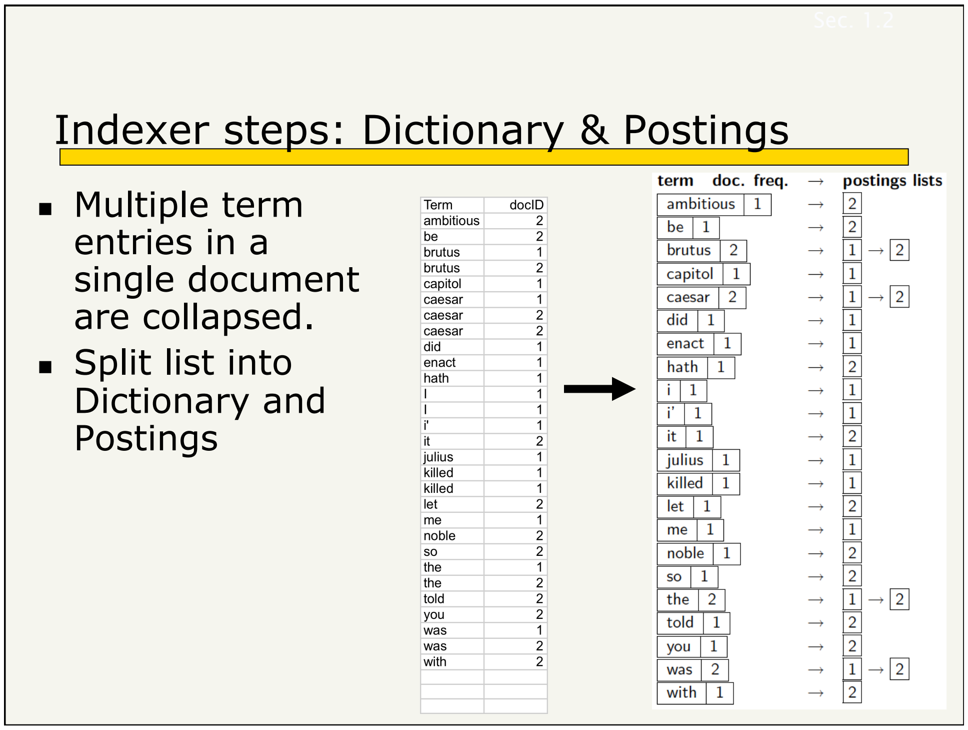
Indexer steps: Sort

- Sort the pairs by terms
  - And then minor sort by 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



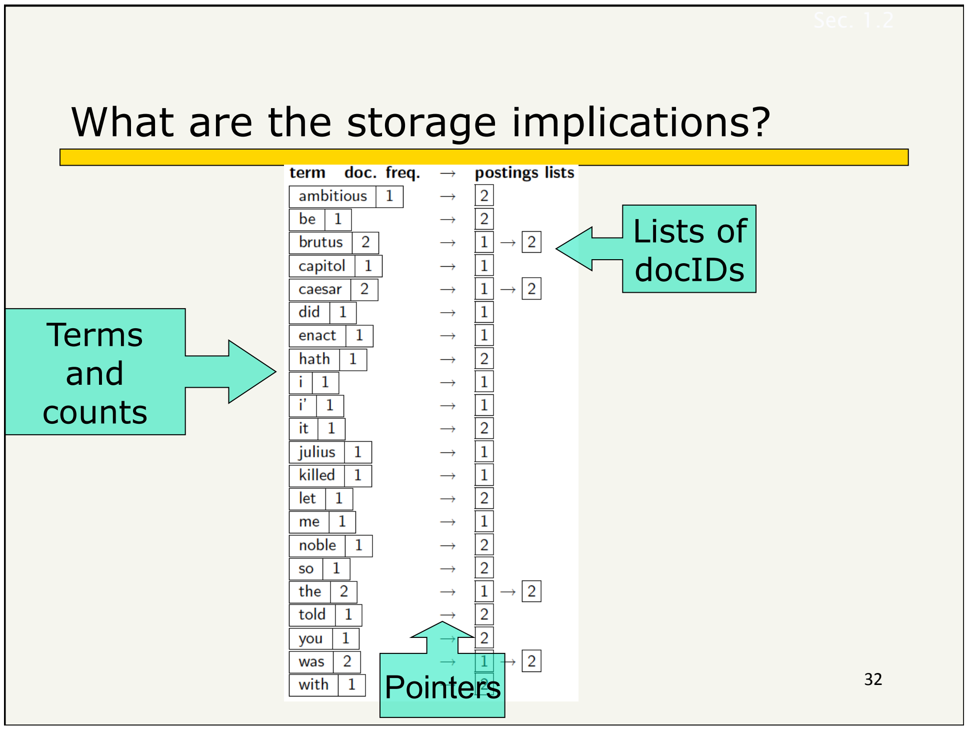
Sec. 1.2

Indexer steps: Dictionary & Postings

- Multiple term entries in a single document are collapsed.
- Split list into Dictionary and Postings

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



Sec. 1.2

What are the storage implications?

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

Lists of docIDs
Terms and counts
Pointers

32

Indexing

Of course you wouldn't really do it that way for large collections. Why?

Too slow and too large:
- The indexer would be too slow
- The resulting index would be too big.

33

Next time

- Read Chapters 1 and 2
- Get Lucene installed on whatever machine you plan to work on

34