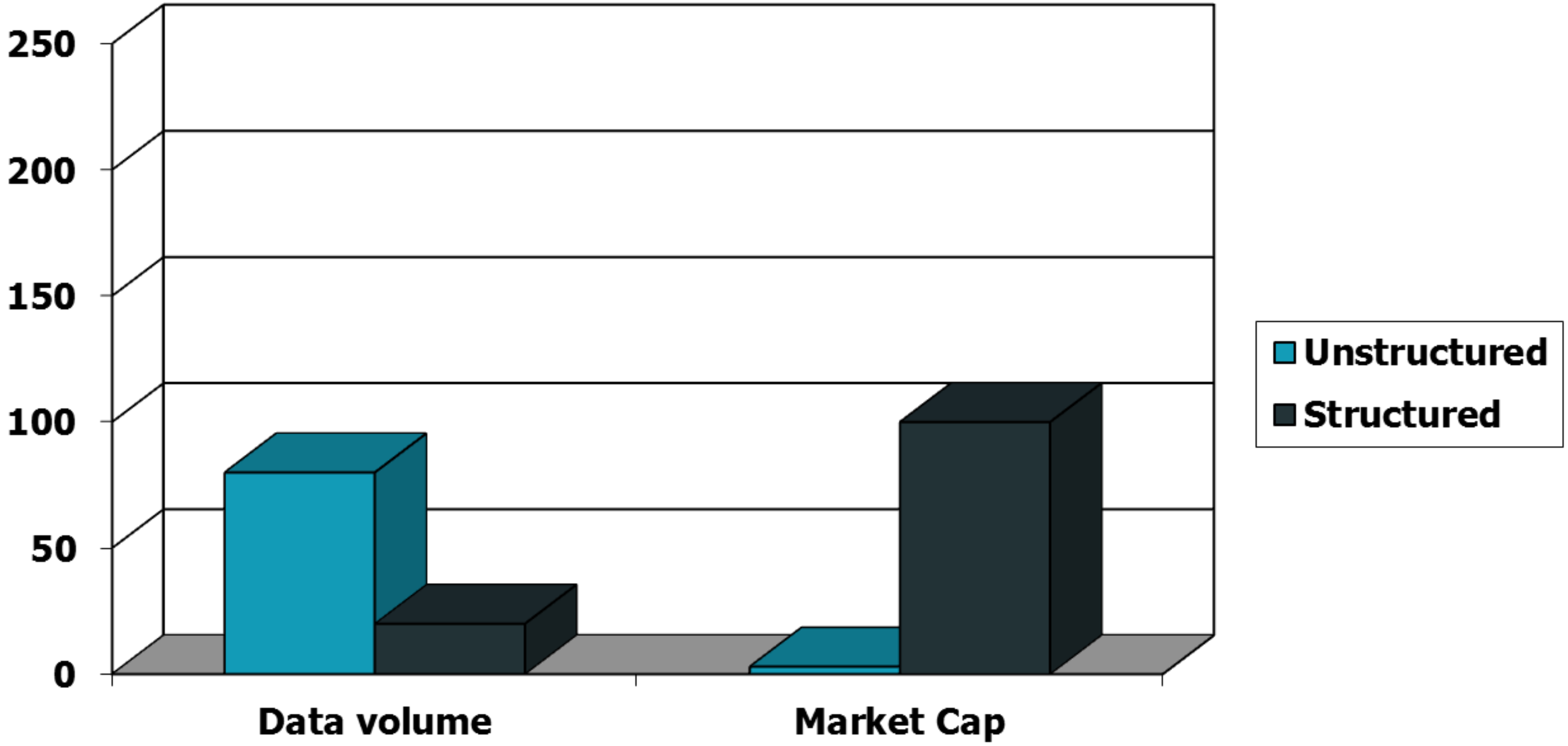
Information Retrieval

SU 5050
LECTURE 2
JESSICA L. MCCARTY, PH.D.

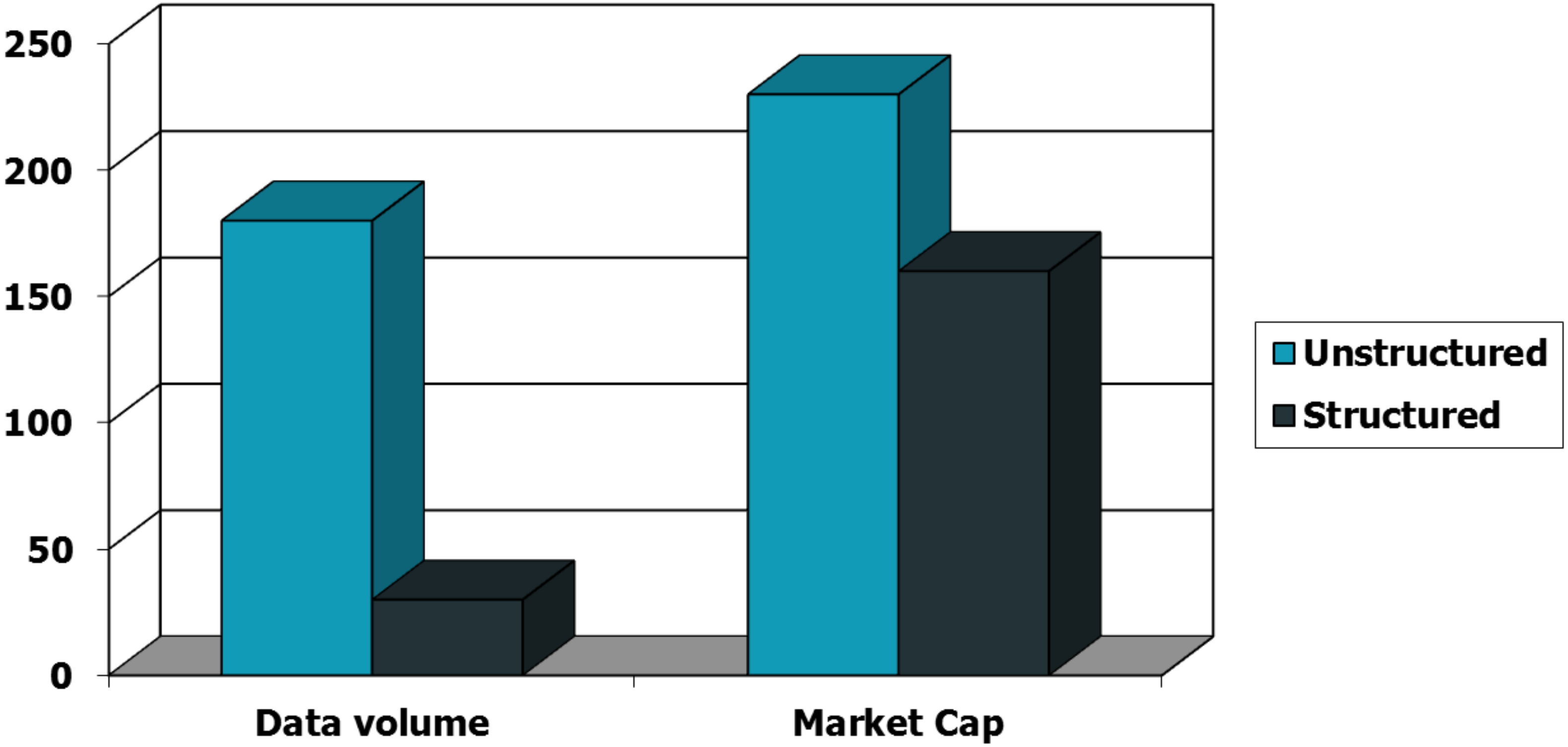
Information Retrieval

- 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).

Unstructured (text) vs. structured (database) data in the mid-1990s



Unstructured (text) vs. structured (database) data circa today

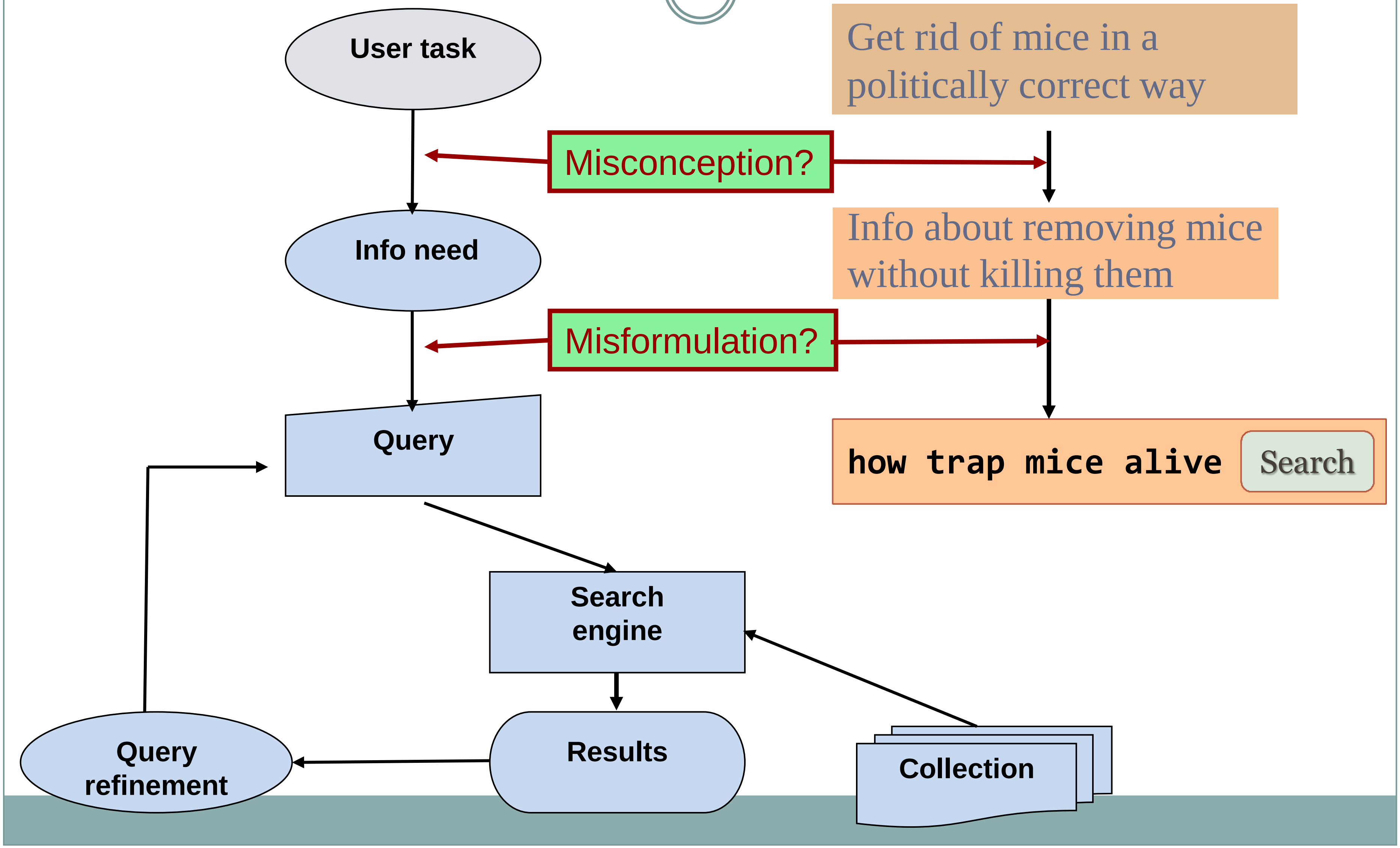


Basic assumptions of Information Retrieval

- Collection: A set of documents
  - Assume it is a ***static*** collection for the moment

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

The classic search model



User task

Info need

Query

Search engine

Results

Query refinement

Collection

Misconception?

Misformulation?

Get rid of mice in a politically correct way

Info about removing mice without killing them

`how trap mice alive` Search

How good are the retrieved docs?

- *Precision* : Fraction of retrieved docs that are relevant to the user's **information need**
- *Recall* : Fraction of relevant docs in collection that are retrieved

Unstructured data in 1620

- 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)

grep is unix/linux command to print lines matching patterns

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***

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.

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.

Not gonna build that matrix

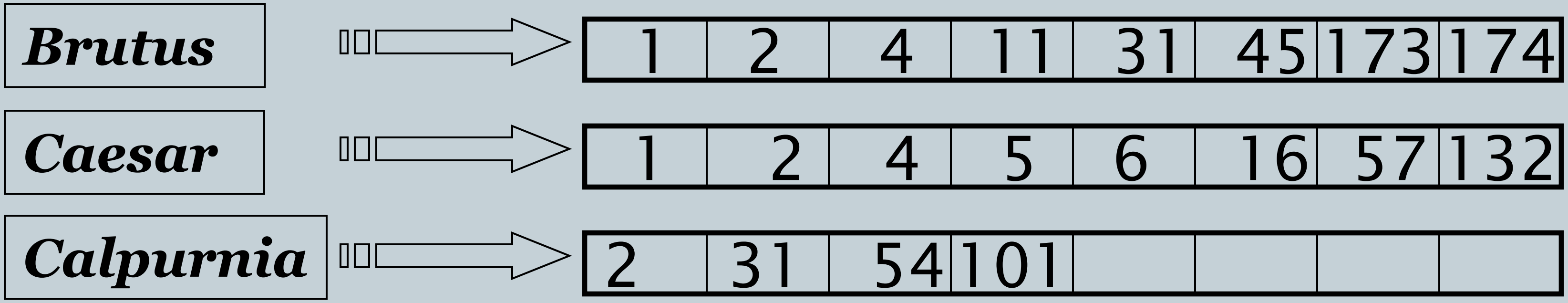
- 500K x 1M matrix has half-a-trillion 0’s and 1’s.

- But it has no more than one billion 1’s.
  - matrix is extremely sparse.

- What’s a better representation?
  - We only record the 1 positions.

Inverted index

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

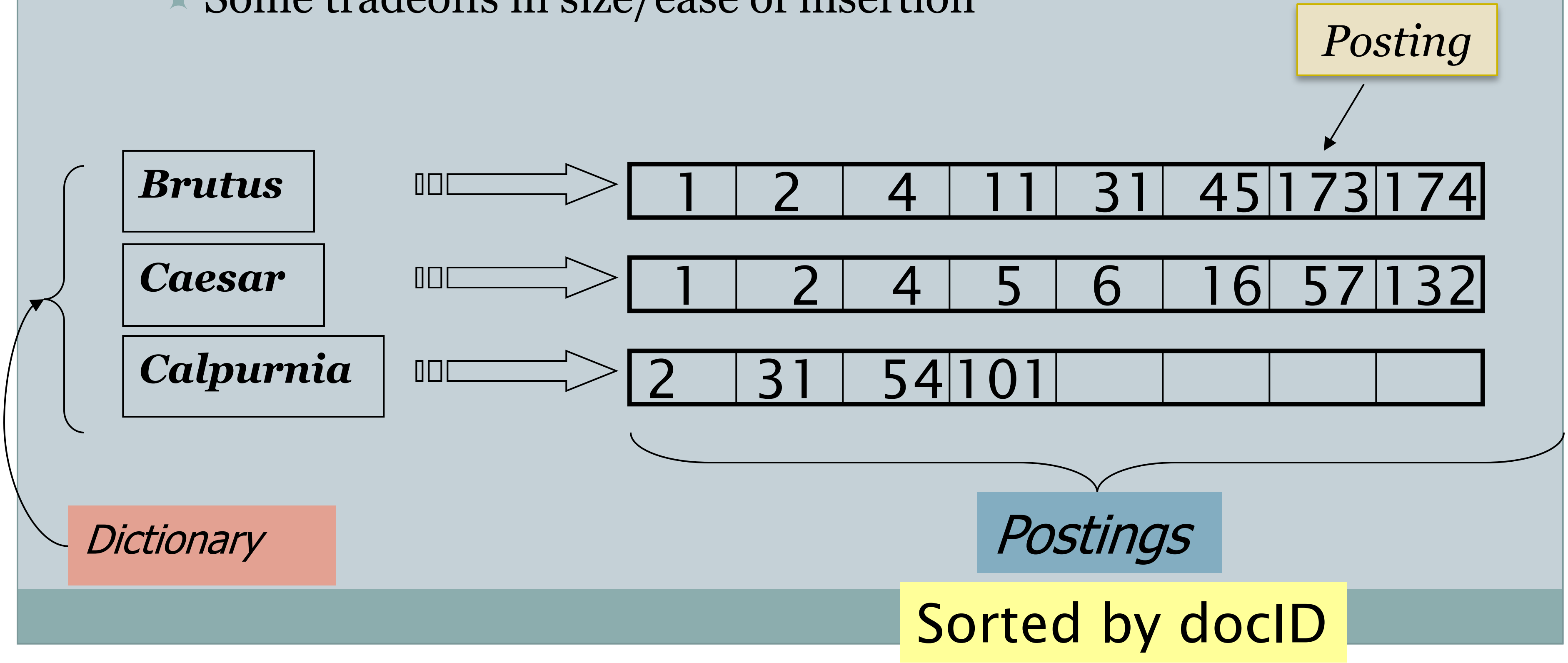


Brutus	1	2	4	11	31	45	173	174
Caesar	1	2	4	5	6	16	57	132
Calpurnia	2	31	54	101				

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



Posting

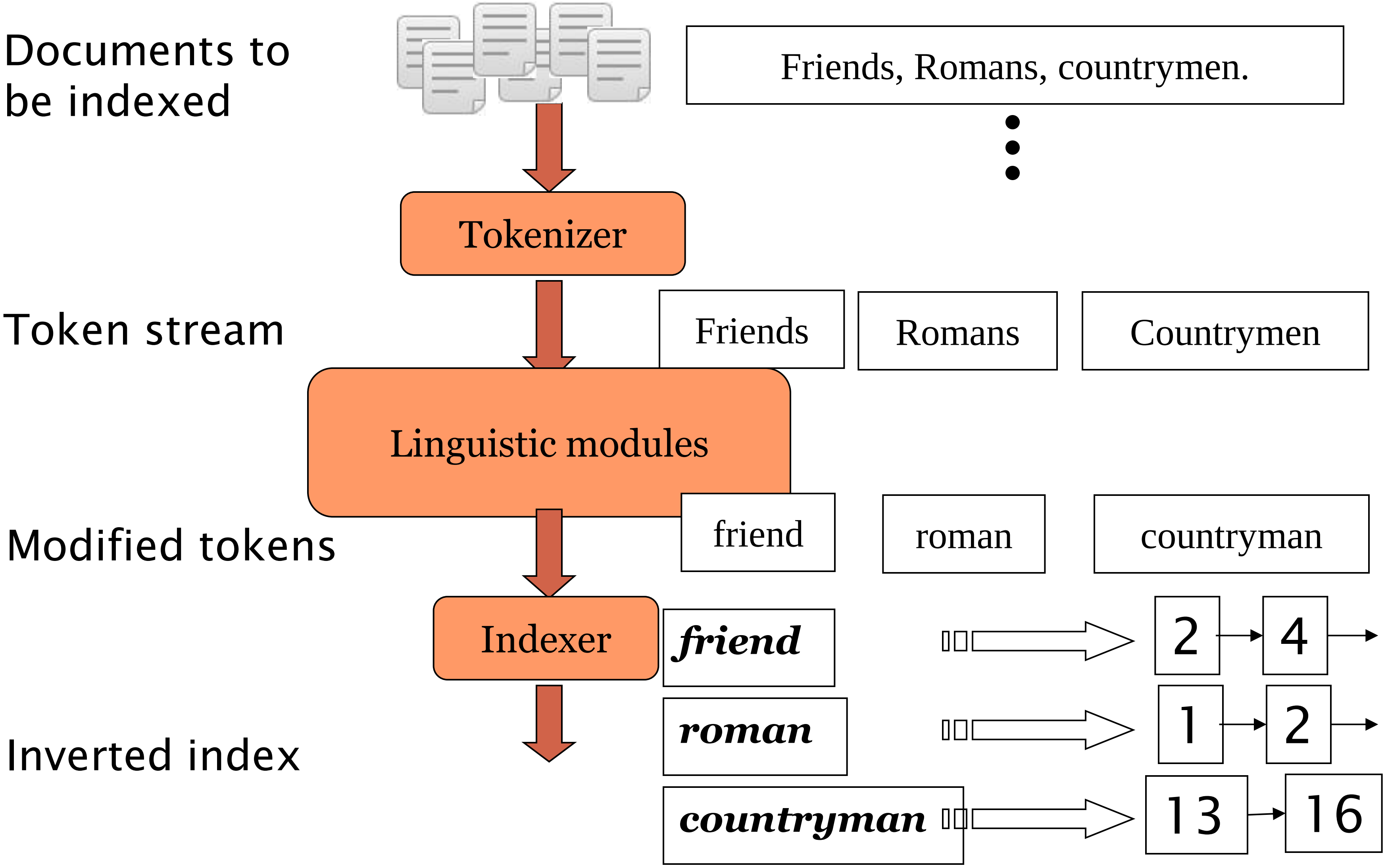
Brutus → 1 2 4 11 31 45 173 174

Caesar → 1 2 4 5 6 16 57 132

Calpurnia → 2 31 54 101

Sorted by docID

Inverted index construction



Documents to be indexed

Friends, Romans, countrymen.

⋮

Tokenizer

Token stream

Friends | Romans | Countrymen

Linguistic modules

Modified tokens

friend | roman | countryman

Indexer

Inverted index

friend → 2 → 4 →

roman → 1 → 2 →

countryman → 13 → 16

Initial stages of text processing

- Tokenization
  - Cut character sequence into word tokens
    - Deal with ***"John's"***, ***a state-of-the-art solution***
- Normalization
  - Map text and query term to same form
    - You want ***U.S.A.*** and ***USA*** to match
- Stemming
  - We may wish different forms of a root to match
    - ***authorize***, ***authorization***
- Stop words
  - We may omit very common words (or not)
    - ***the, a, to, of***

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

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

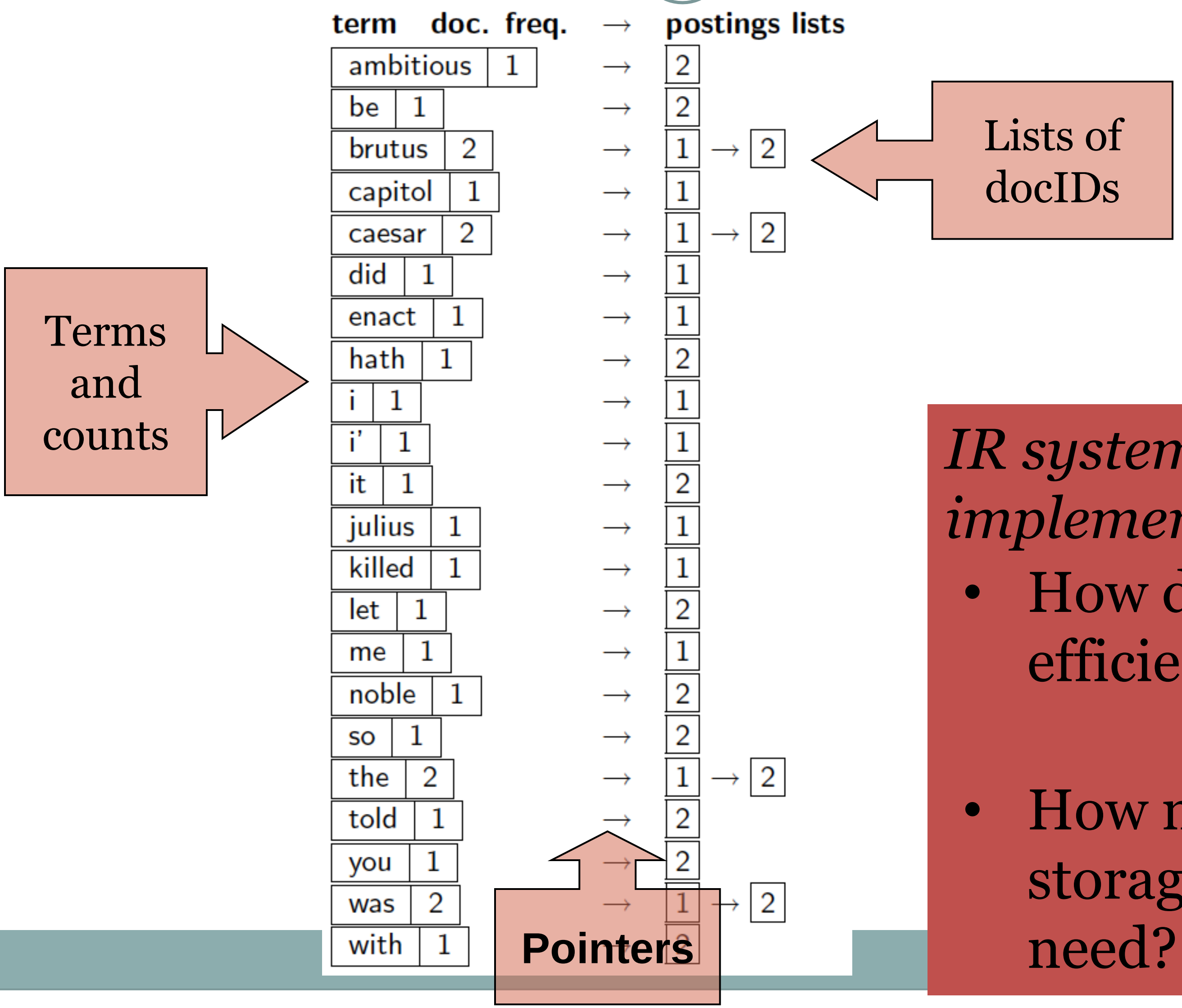
- Multiple term entries in a single document are merged.
- Split into Dictionary and Postings
- Doc. frequency information is added.

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

What about data storage?



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

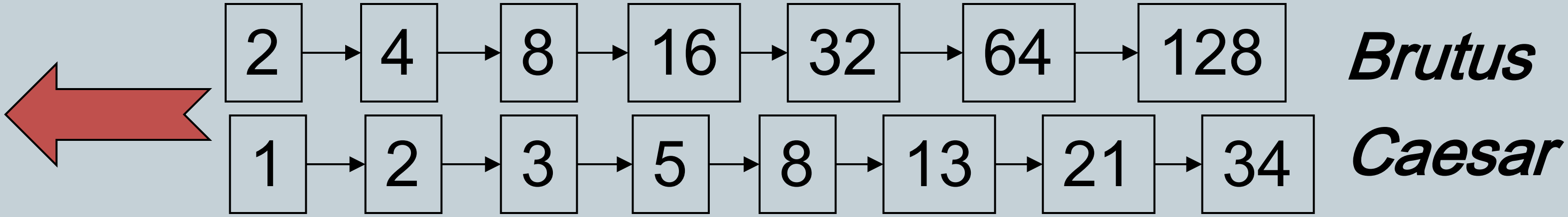
Pointers

IR system implementation

- How do we index efficiently?
- How much storage do we need?

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 (intersect the document sets):

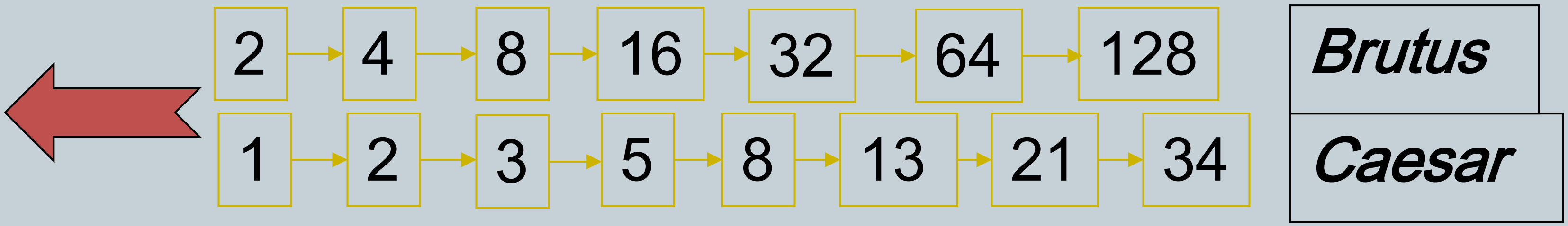


2 → 4 → 8 → 16 → 32 → 64 → 128 ***Brutus***

1 → 2 → 3 → 5 → 8 → 13 → 21 → 34 ***Caesar***

The merge

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



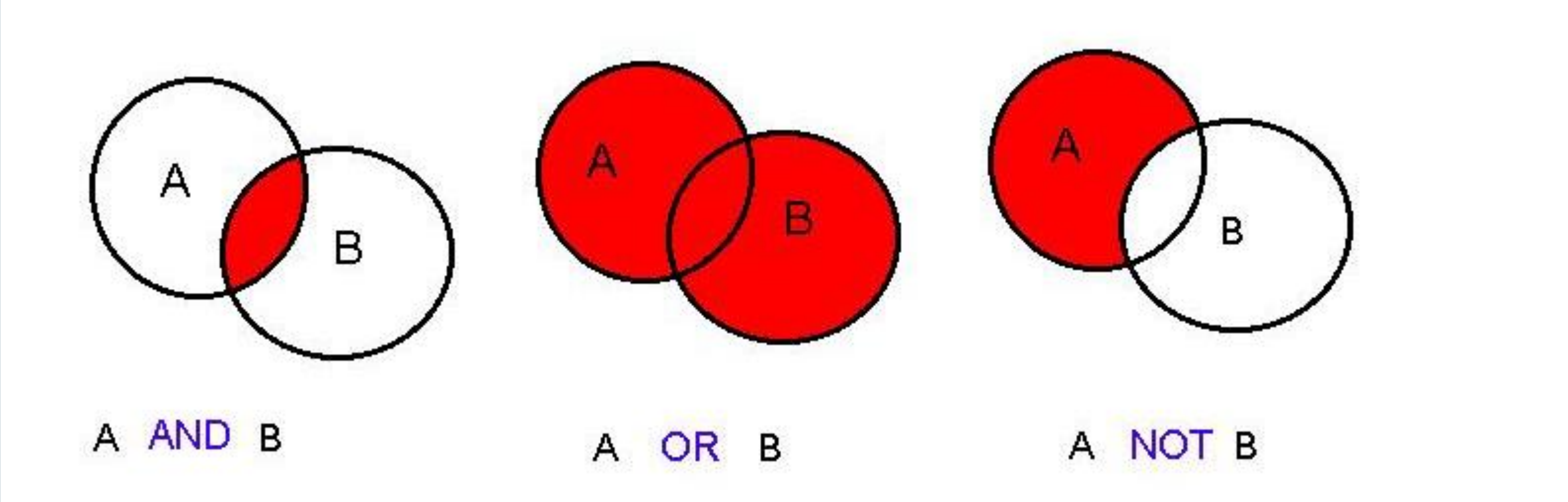
If the list lengths are *x* and *y*, the merge takes O(*x*+*y*) operations.

Crucial: postings sorted by docID.

Boolean queries: Exact match

- The Boolean retrieval model is being able to ask a query that is a Boolean expression:
  - Boolean Queries are queries using *AND*, *OR* and *NOT* to join query terms
    - Views each document as a set of words
    - Is precise: document matches condition or not.
  - Perhaps the simplest model to build an IR system on
- Primary commercial retrieval tool for 3 decades.
- Many search systems you still use are Boolean:
  - Email, library catalog, Mac OS X Spotlight, LaTeX

Boolean Operators



Example: WestLaw http://www.westlaw.com/

- Largest commercial (paying subscribers) legal search service (started 1975; ranking added 1992; new federated search added 2010)
- Tens of terabytes of data; ~700,000 users
- Majority of users *still* use Boolean queries
- Example query:
  - What is the statute of limitations in cases involving the federal tort claims act?
  - LIMIT! /3 STATUTE ACTION /S FEDERAL /2 TORT /3 CLAIM
    - /3 = within 3 words, /S = in same sentence

Boolean queries: More general merges

- Exercise: Adapt the merge for the queries:

  Brutus *AND NOT* ***Caesar***

  Brutus *OR NOT* ***Caesar***

- Can we still run through the merge in time O(x+y)? What can we achieve?

Merging

What about an arbitrary Boolean formula?

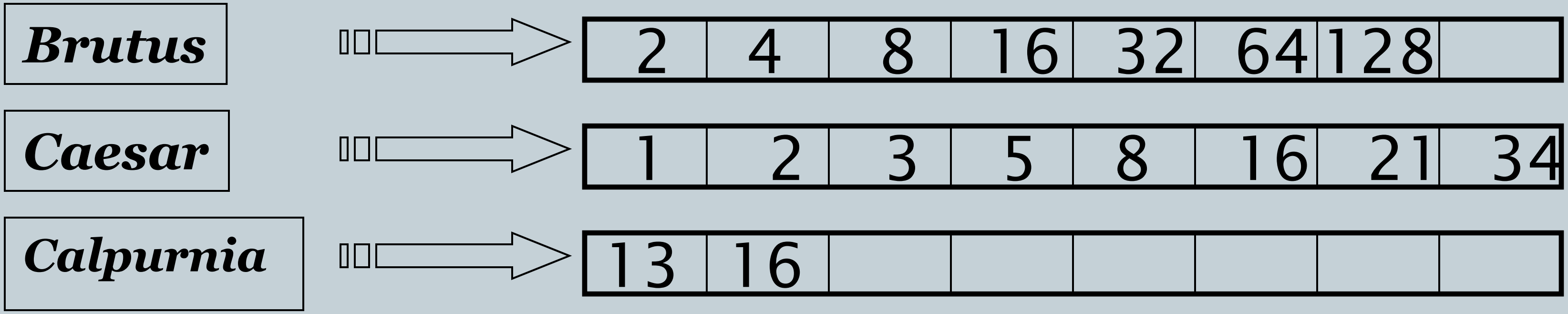
(Brutus *OR* ***Caesar)*** *AND NOT*

(Antony *OR* ***Cleopatra)***

- Can we always merge in “linear” time?
  - Linear in what?
- Can we do better?

Query optimization

- What is the best order for query processing?
- Consider a query that is an *AND* of *n* terms.
- For each of the *n* terms, get its postings, then *AND* them together.



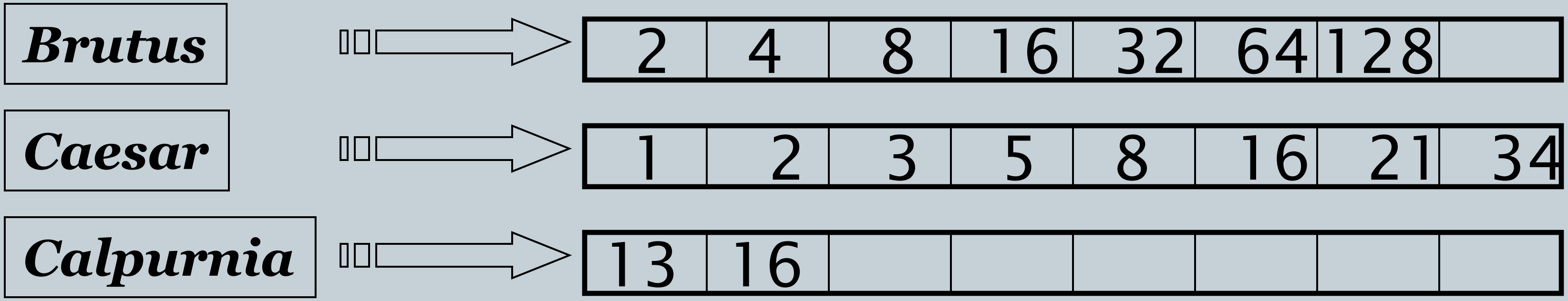
Brutus	2	4	8	16	32	64	128	
Caesar	1	2	3	5	8	16	21	34
Calpurnia	13	16						

Query: ***Brutus*** *AND* ***Calpurnia*** *AND* ***Caesar***

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



Brutus	2	4	8	16	32	64	128	
Caesar	1	2	3	5	8	16	21	34
Calpurnia	13	16						

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

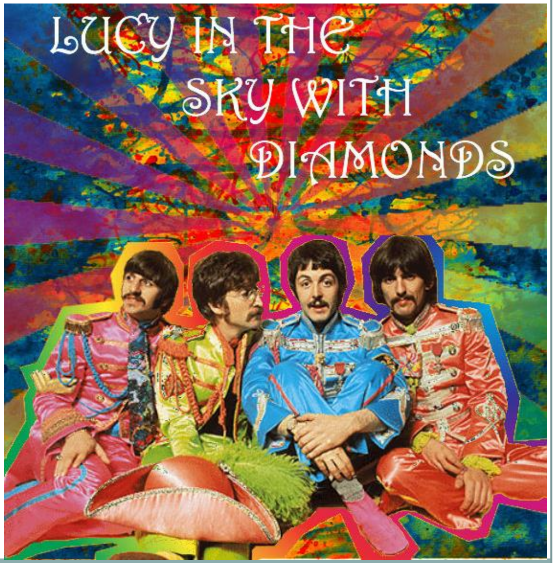
Quick Exercise

- Recommend a query processing order for

(tangerine** OR **trees)** AND **(marmalade** OR **skies)** AND **(kaleidoscope** OR **eyes)

- Which two terms should we process first?

Term	Freq
eyes	213312
kaleidoscope	87009
marmalade	107913
skies	271658
tangerine	46653
trees	316812



Phrase queries

- We want to be able to answer queries such as ***"michigan tech university"*** – as a phrase
- Thus the sentence *"I went to undergrad in Houghton"* is not a match.
  - The concept of phrase queries has proven easily understood by users; one of the few "advanced search" ideas that works
  - Many more queries are *implicit phrase queries*
- For this, it no longer suffices to store only <*term* : *docs*> entries

A first attempt: Biword indexes

- Index every consecutive pair of terms in the text as a phrase
- For example the text “Friends, Romans, Countrymen” would generate the biwords
  - ***friends romans***
  - ***romans countrymen***
- Each of these biwords is now a dictionary term
- Two-word phrase query-processing is now immediate.

Longer phrase queries

- Longer phrases can be processed by breaking them down
- ***michigan tech university houghton*** can be broken into the Boolean query on biwords:

michigan tech *AND* ***university houghton*** *AND* ***tech houghton***

Without the docs, we cannot verify that the docs matching the above Boolean query do contain the phrase.

Can have false positives!

Issues for biword indexes

- False positives, as noted before
- Index blowup due to bigger dictionary
  - Infeasible for more than biwords, big even for them
- Biword indexes are not the standard solution (for all biwords) but can be part of a compound strategy

Positional index size

- Need an entry for each occurrence, not just once per document
- Index size depends on average document size
  - Average web page has <1000 terms
  - SEC filings, books, even some epic poems … easily 100,000 terms
- Consider a term with frequency 0.1%

Document size	Postings	Positional postings
1000	1	1
100,000	1	100

Rules of thumb

- A positional index is 2–4 as large as a non-positional index
- Positional index size 35–50% of volume of original text
  - **Caveat: all of this holds for “English-like” languages**

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

Unstructured data

- Typically refers to free 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
- Twitter, facebook, instagram, all social media

Semi-structured data

- In fact almost no data is “unstructured”
- E.g., this slide has distinctly identified zones such as the *Title* and *Bullets*
    - … to say nothing of linguistic structure
- Facilitates “semi-structured” search such as
    - *Title* contains data AND *Bullets* contain search
- Or even
    - *Title* is about Object Oriented Programming AND *Author* something like stro*rup
    - where * is the wild-card operator

Introduction to Metadata

What is metadata?

Metadata is… data about data!

(from the Greek preposition μετά meaning "after” or "with”)

Basically, metadata is any kind of information that describes something else.

The need for metadata

- Sufficiency
  - Can an object describe itself?
    - E.g., images
- Scalability
  - Allows for rapid searching
    - Searching metadata fields vs. large data files
- Interoperability
  - Can exchange data using mutually agreed metadata formats

Different ways of thinking about metadata

- *Authoritative* vs. *user-created*
- Different *types* of metadata to describe various aspects of the same thing
- Ontologies (nature of being), taxonomies, vocabularies
- Metadata *standards* and *formats*

Authoritative metadata

- AKA ‘top-down’
- Created by project team
- Formalized; focus on control
- Specialists in (at least one aspect of) the field
- Focus and coverage will depend on the requirements of the project and agency

User-created metadata

- AKA ‘bottom-up’
- Social tagging
  - May be open or within a community
- Less focused; what the “tagging public” sees
- Generally less structured, not prescriptive

Types of metadata

- **Descriptive:** Facilitates discovery and describes intellectual content
- **Administrative:** Facilitates management of digital and analog resources
- **Technical:** Describes the technical aspects
- **Structural:** Describes the relationships within object
- **Preservation:** Supports long-term retention and may overlap with technical, administrative, and structural metadata

Descriptive metadata

It is always necessary to differentiate between the description of…

- *Content*
- *Source (if there is one!)*
- *Digital file/object*

For *born-digital* objects, the digital object *is* the source

Administrative metadata

- Facilitates management of files
- Describes the creation/derivation of files
  - Responsible Individuals and institutions
  - Dates
  - Locations
- Technical specifications (e.g., file size, file format)

Structural metadata

- Describes/defines relationships between and among files
  - describing *collections, versions*
  - describing *projects*

Relationships are usually, but need not be, 1:1

Identifying what collection or project a file belongs to

Identifying what files belong to which collection or project

Identifying what project a collection belongs to

Ontologies, taxonomies, controlled vocabularies

- Controlled vocabulary: a list of terms

- Taxonomy: a collection of controlled vocabulary terms organized into a hierarchical structure

- Ontology: a formal representation of a set of concepts within a domain, and the relationships between these concepts

Controlled vocabulary

Internet Assigned Numbers Authority Language Subtag Registry (http://www.iana.org/assignments/language-subtag-registry)

- Controlled list of explicitly enumerated terms
- Unambiguous definitions for each term

1. If the same term is commonly used to mean different concepts in different contexts, then its name is explicitly qualified to resolve this ambiguity.
2. If multiple terms are used to mean the same thing, one of the terms is identified as the preferred term in the controlled vocabulary and the other terms are listed as synonyms or aliases.

Metadata *standards* and *formats*

The first questions of metadata:

- What do we want to describe?
- How to we want to describe it?

Using accepted *standards*, expressed in widely-used or easily mapped *formats*, will ensure that our metadata is accessible.

Metadata standards

- Standards are widely-used (hence *standard)* prescriptive recommendations guiding
  - Defining fields: “name” “title” “identifier” “subject” “physicalDescription” “location”
  - Structure and hierarchy within the metadata itself
  - Controlled vocabularies

Metadata formats

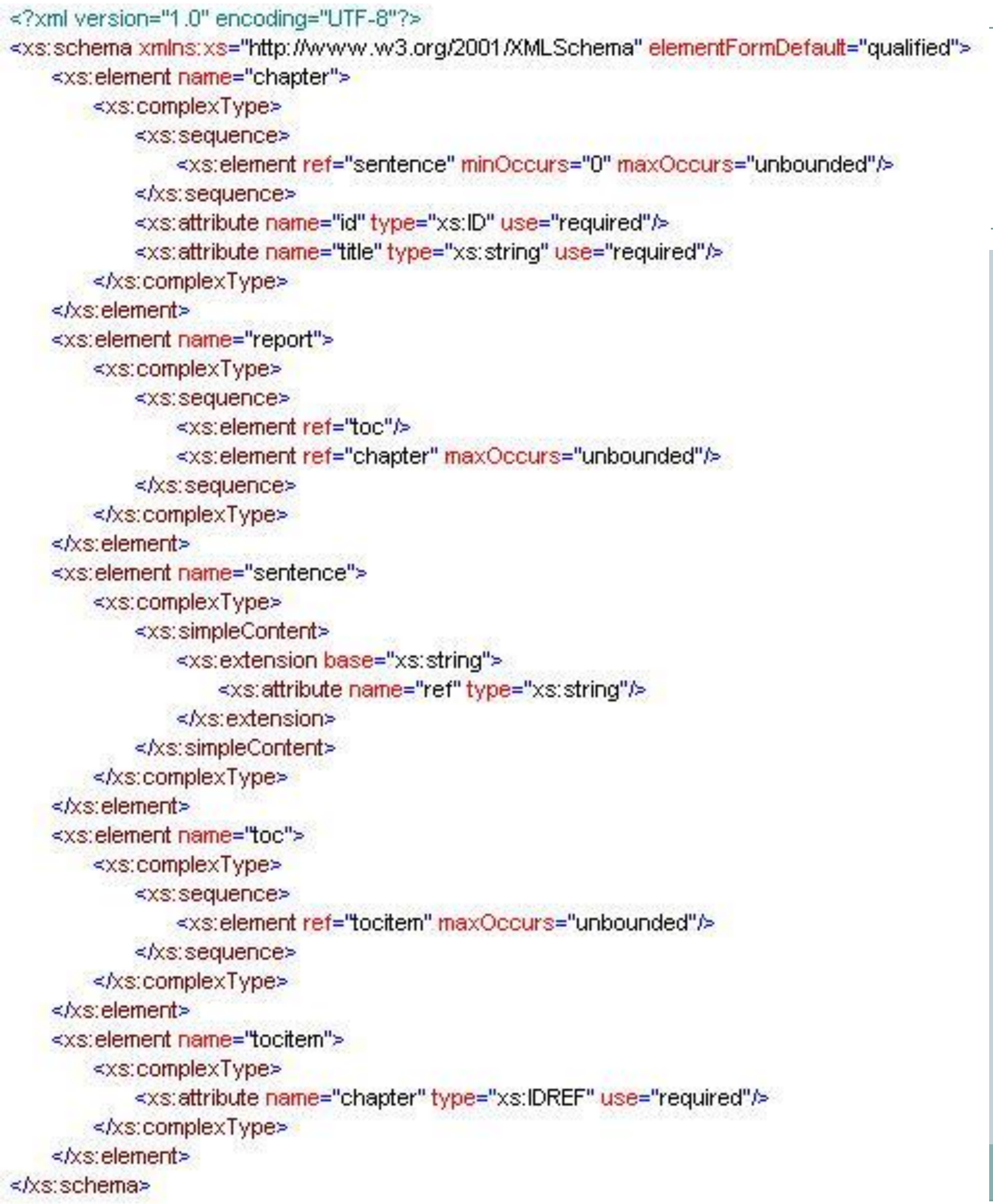
- Extensible Markup Language (XML)
  - Allows for combining and interoperability
  - XML flexibility

- Any other conceivable format
  - MS Word? PDF? Post-it notes?

  - Excel, FileMaker Pro, Access DB, CSV

XML

- Extensible Markup Language
  - Uses tags to describe data elements
- Defined by W3c (World Wide Web Consortium)
  - For data exchange over networks
- Few predefined elements
  - Minimalist
- Tree structure
  - Parent nodes, subnodes



```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="chapter">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sentence" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:ID" use="required"/>
            <xs:attribute name="title" type="xs:string" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="report">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="toc"/>
                <xs:element ref="chapter" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="sentence">
        <xs:complexType>
            <xs:simpleContent>
                <xs:extension base="xs:string">
                    <xs:attribute name="ref" type="xs:string"/>
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="toc">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="tocitem" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="tocitem">
        <xs:complexType>
            <xs:attribute name="chapter" type="xs:IDREF" use="required"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

XML

- Elements in XML described by a DTD
  - Document Type Definition
- Describe elements
  - Name
  - Type
  - Format
  - Order

XML

- XML is **well-formed** if
  - Element tags are matched
  - The tags are closed correctly

- XML document is **valid** if
  - Structure conforms to the DTD

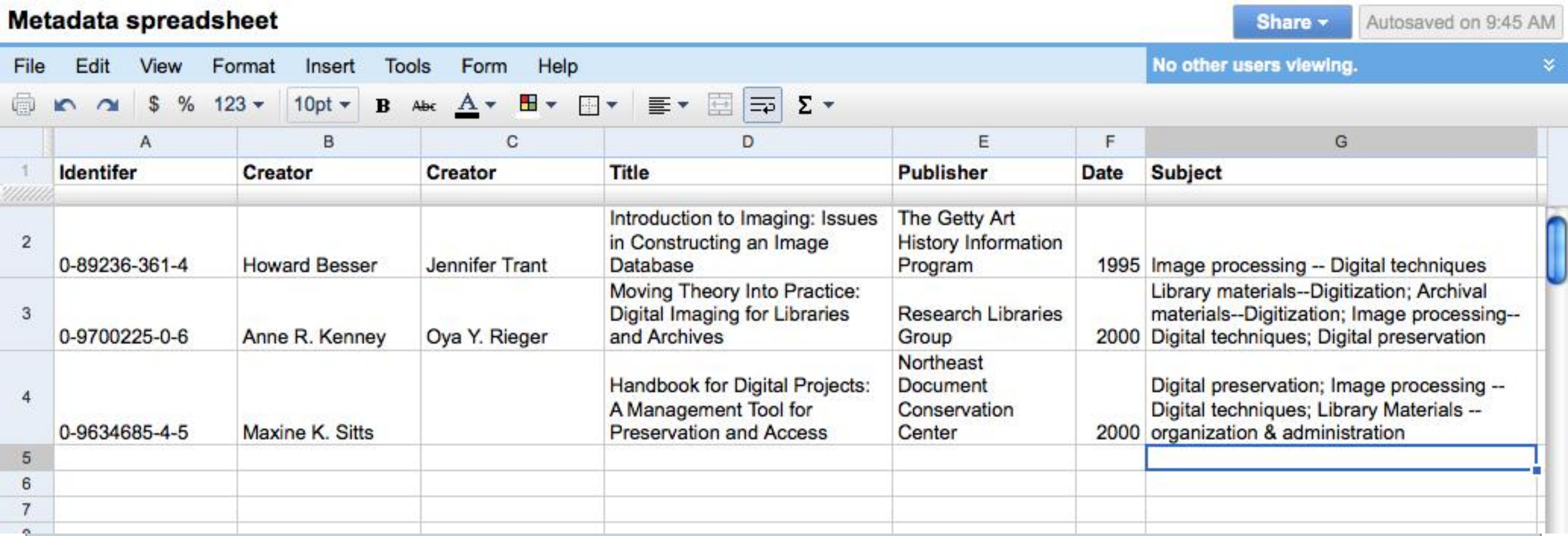
Metadata and XML

- The elements in an XML document describe the data
  - Elements are metadata
- Grammar in the DTD describe the elements
  - Metadata for the elements
- The flexibility of the XML illustrates the use of metadata

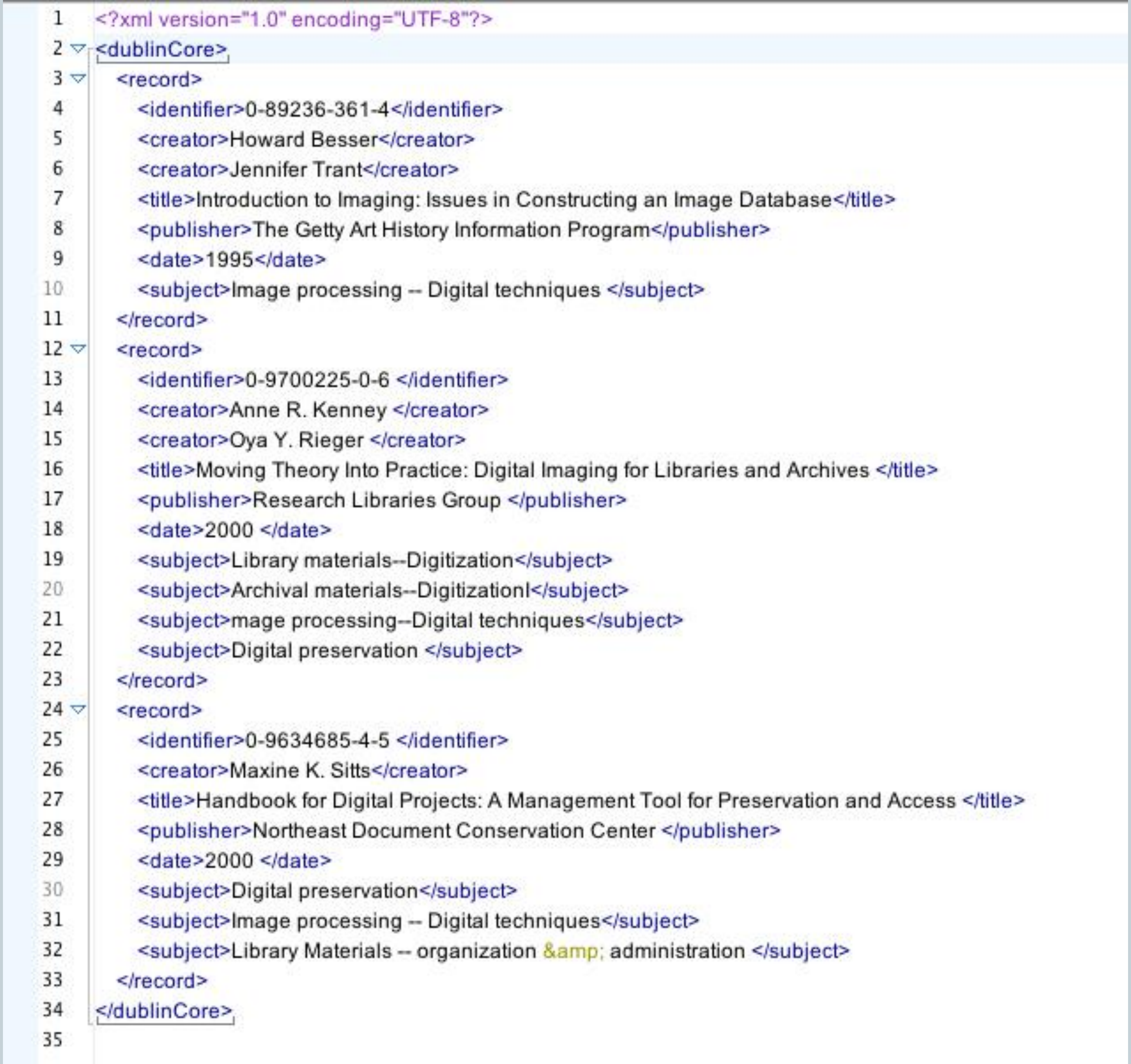
Problems with XML

- Large files
  - Tags add bulk
  - Addressed with compression
- Security
  - Files are plain text
  - Addressed with encryption

In some but not all cases, the *semantics* of metadata is separate from the *format* of metadata



	A	B	C	D	E	F	G
1	Identifer	Creator	Creator	Title	Publisher	Date	Subject
2	0-89236-361-4	Howard Besser	Jennifer Trant	Introduction to Imaging: Issues in Constructing an Image Database	The Getty Art History Information Program	1995	Image processing -- Digital techniques
3	0-9700225-0-6	Anne R. Kenney	Oya Y. Rieger	Moving Theory Into Practice: Digital Imaging for Libraries and Archives	Research Libraries Group	2000	Library materials--Digitization; Archival materials--Digitization; Image processing--Digital techniques; Digital preservation
4	0-9634685-4-5	Maxine K. Sitts		Handbook for Digital Projects: A Management Tool for Preservation and Access	Northeast Document Conservation Center	2000	Digital preservation; Image processing -- Digital techniques; Library Materials -- organization & administration
5							
6							
7							



```xml
<?xml version="1.0" encoding="UTF-8"?>
<dublinCore>
  <record>
    <identifier>0-89236-361-4</identifier>
    <creator>Howard Besser</creator>
    <creator>Jennifer Trant</creator>
    <title>Introduction to Imaging: Issues in Constructing an Image Database</title>
    <publisher>The Getty Art History Information Program</publisher>
    <date>1995</date>
    <subject>Image processing -- Digital techniques </subject>
  </record>
  <record>
    <identifier>0-9700225-0-6 </identifier>
    <creator>Anne R. Kenney </creator>
    <creator>Oya Y. Rieger </creator>
    <title>Moving Theory Into Practice: Digital Imaging for Libraries and Archives </title>
    <publisher>Research Libraries Group </publisher>
    <date>2000 </date>
    <subject>Library materials--Digitization</subject>
    <subject>Archival materials--DigitizationI</subject>
    <subject>mage processing--Digital techniques</subject>
    <subject>Digital preservation </subject>
  </record>
  <record>
    <identifier>0-9634685-4-5 </identifier>
    <creator>Maxine K. Sitts</creator>
    <title>Handbook for Digital Projects: A Management Tool for Preservation and Access </title>
    <publisher>Northeast Document Conservation Center </publisher>
    <date>2000 </date>
    <subject>Digital preservation</subject>
    <subject>Image processing -- Digital techniques</subject>
    <subject>Library Materials -- organization &amp; administration </subject>
  </record>
</dublinCore>
```

Metadata mapping

- Moving metadata from one standard/format to another standard/format
- Not always pretty…
- *Important: base the design of your project metadata on an existing standard, and plan it out ahead of time!*

How much is enough?

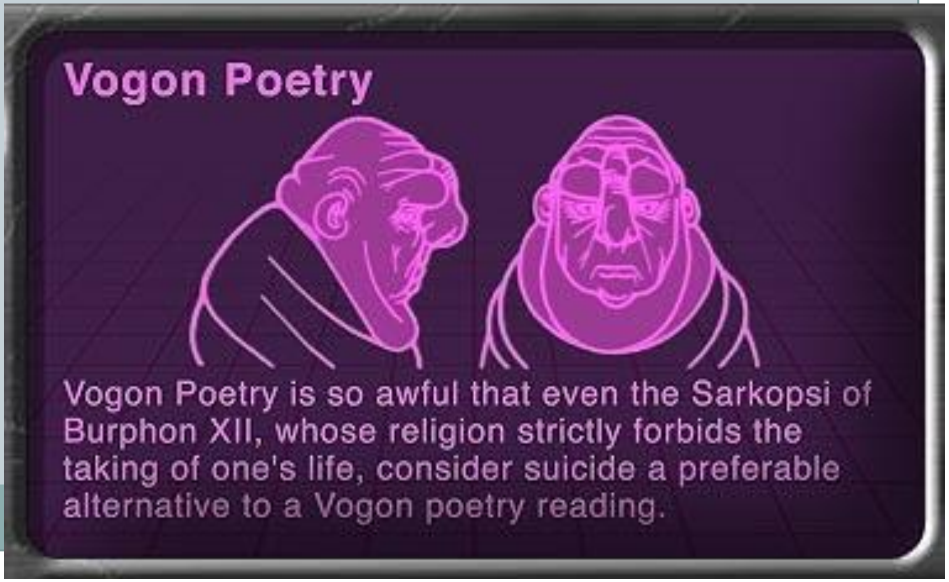
Standards are great! → ***ROBUST***

Just because it's there doesn't mean you have to use it!

Open Access (not magic, a bit scary, but very useful)

Geospatial Metadata

- FDGC (Federal Geographic Data Committee)
  - www.fdgc.gov
- ISO (International Organization for Standardization)
  - http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=53798



Vogon Poetry

Vogon Poetry is so awful that even the Sarkopsi of Burphon XII, whose religion strictly forbids the taking of one's life, consider suicide a preferable alternative to a Vogon poetry reading.