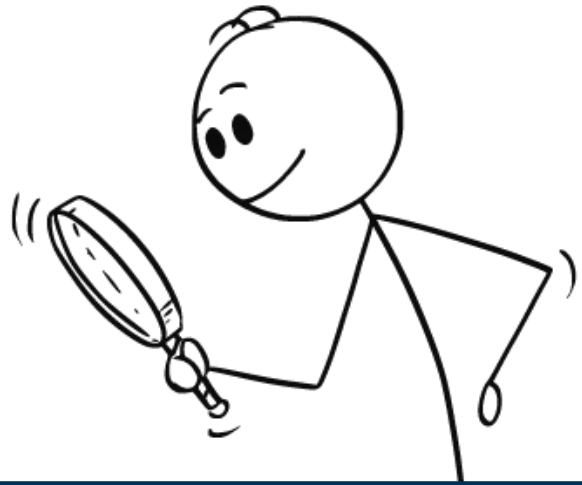


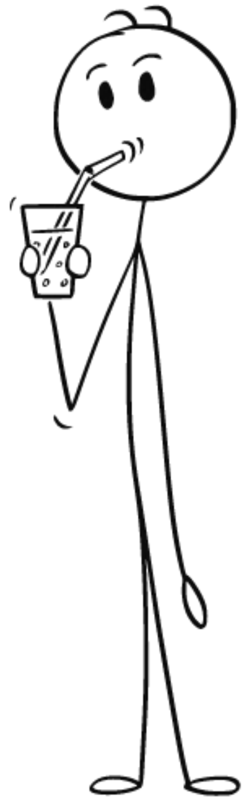
CSE 4/535 Information Retrieval

Sayantan Pal
PhD Student, Department of CSE
338Z Davis Hall

University at Buffalo
Department of CSE

Before we start

1. AI Quiz will be released this Friday and will be due next Friday.
2. Is everything working?
   a. Piazza? Website? Brightspace?
   b. Is the recording visible?
3. Let me know if you want me to slow down. Unless you speak I will not understand you.
4. Remind me when last 10 mins will be left (If I had not finished already)



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)

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)

These days we frequently think first of web search, but there are many other cases (examples):

1. Email search, Searching your laptop
2. Corporate knowledge bases
3. Medical, Legal information retrieval
4. Product Search
5. Social media search

Structured vs Unstructured Data (IR vs DB)

Structured data tends to refer to information in “**tables**”

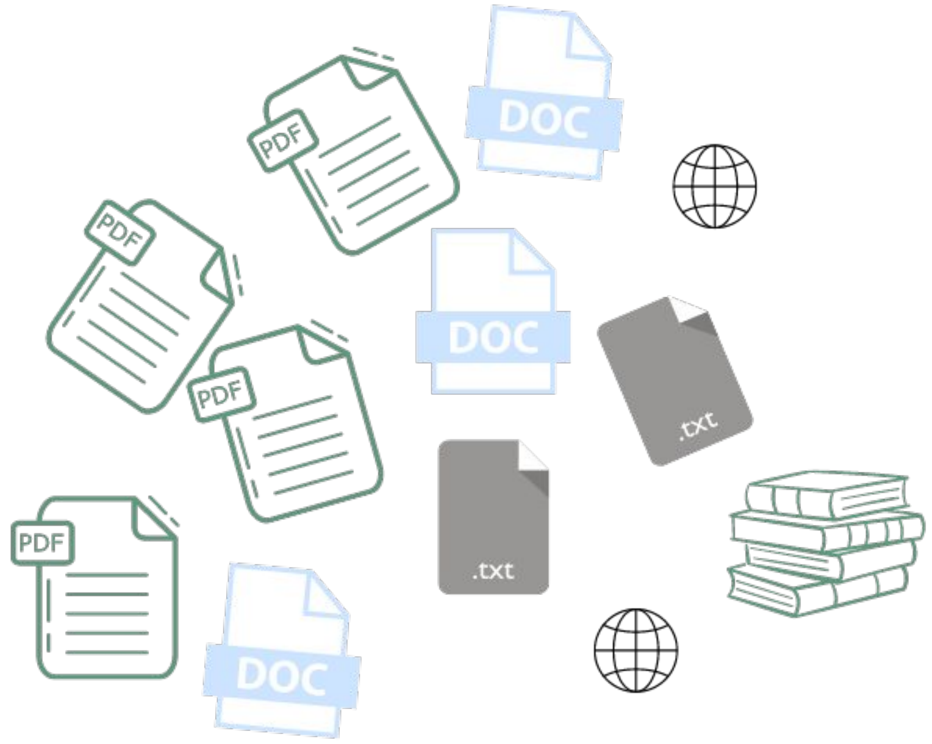
Employee	**Manager**	**Salary**
Smith	Jones	50000
Chang	Smith	60000
Ivy	Smith	50000

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



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

The Importance of Converting Unstructured to Semi-Structured Data in Information Retrieval (IR)

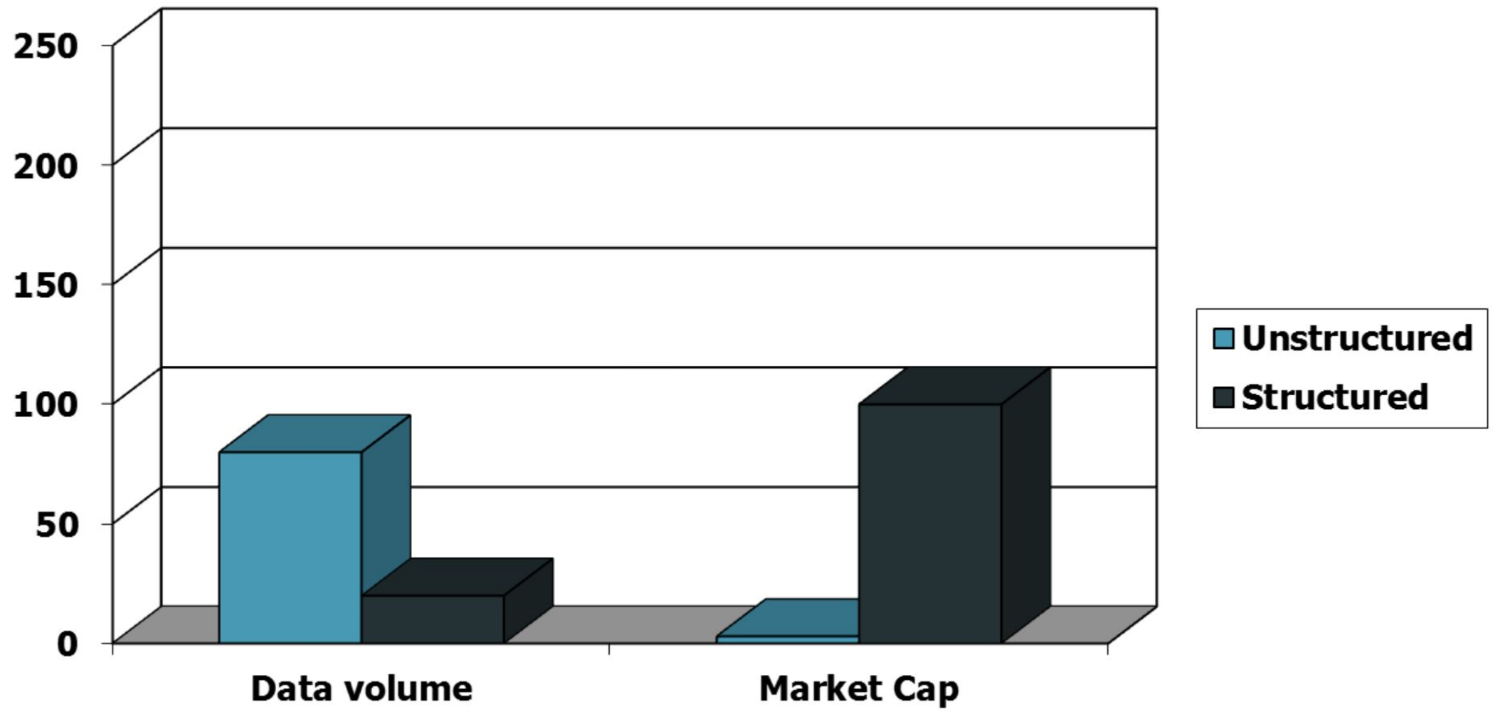
Why Conversion is Necessary:

1. **Searchability**: Converting data into a semi-structured format improves its searchability, making it easier for IR systems to find relevant information.
2. **Efficiency**: Semi-structured data is easier to analyze, requiring less computational power than unstructured data, thereby speeding up IR processes.

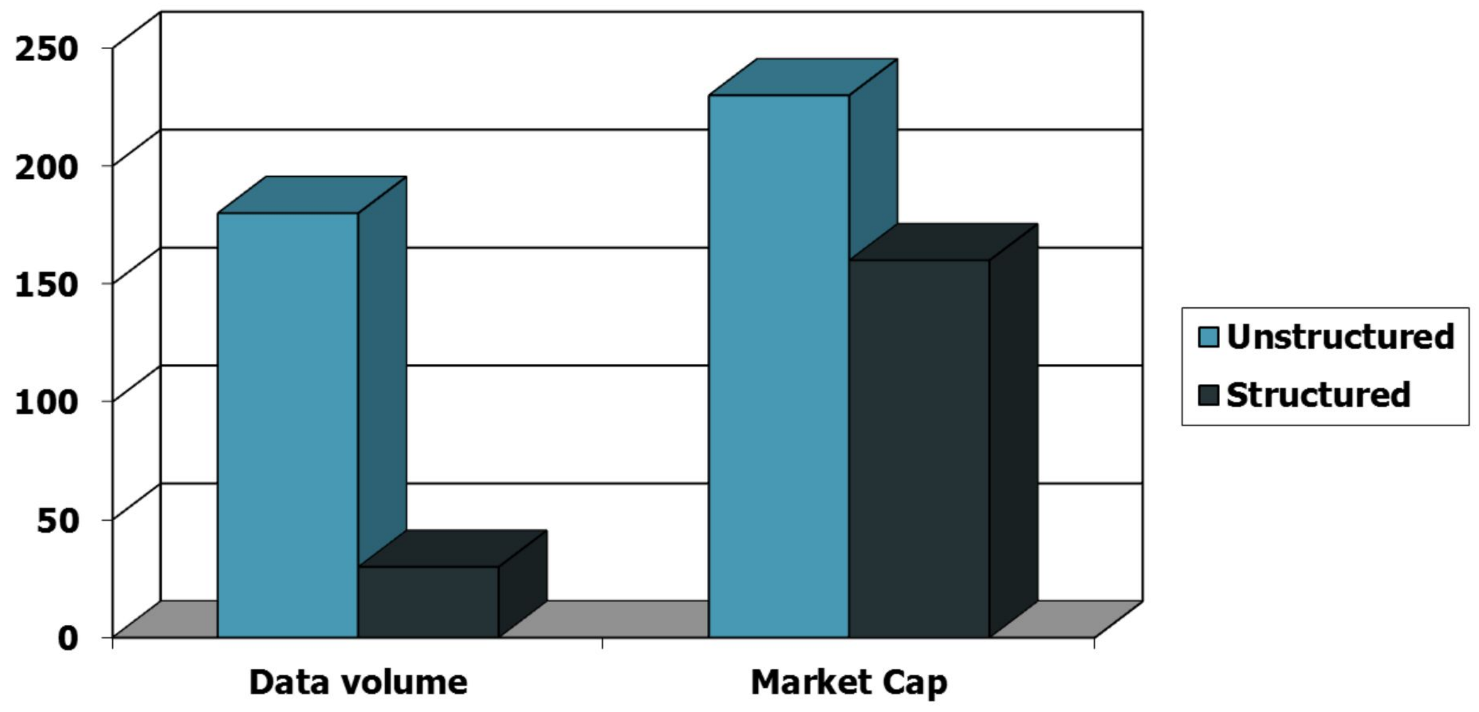
Key Benefits:

1. **Improved Querying**: Tags and hierarchies allow for complex queries, enabling users to find more precisely.
2. **Scalability**: Semi-structured data can easily be scaled
3. **Metadata Utilization**: The presence of metadata means IR systems can understand the context of the data, resulting in more accurate search results.

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



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

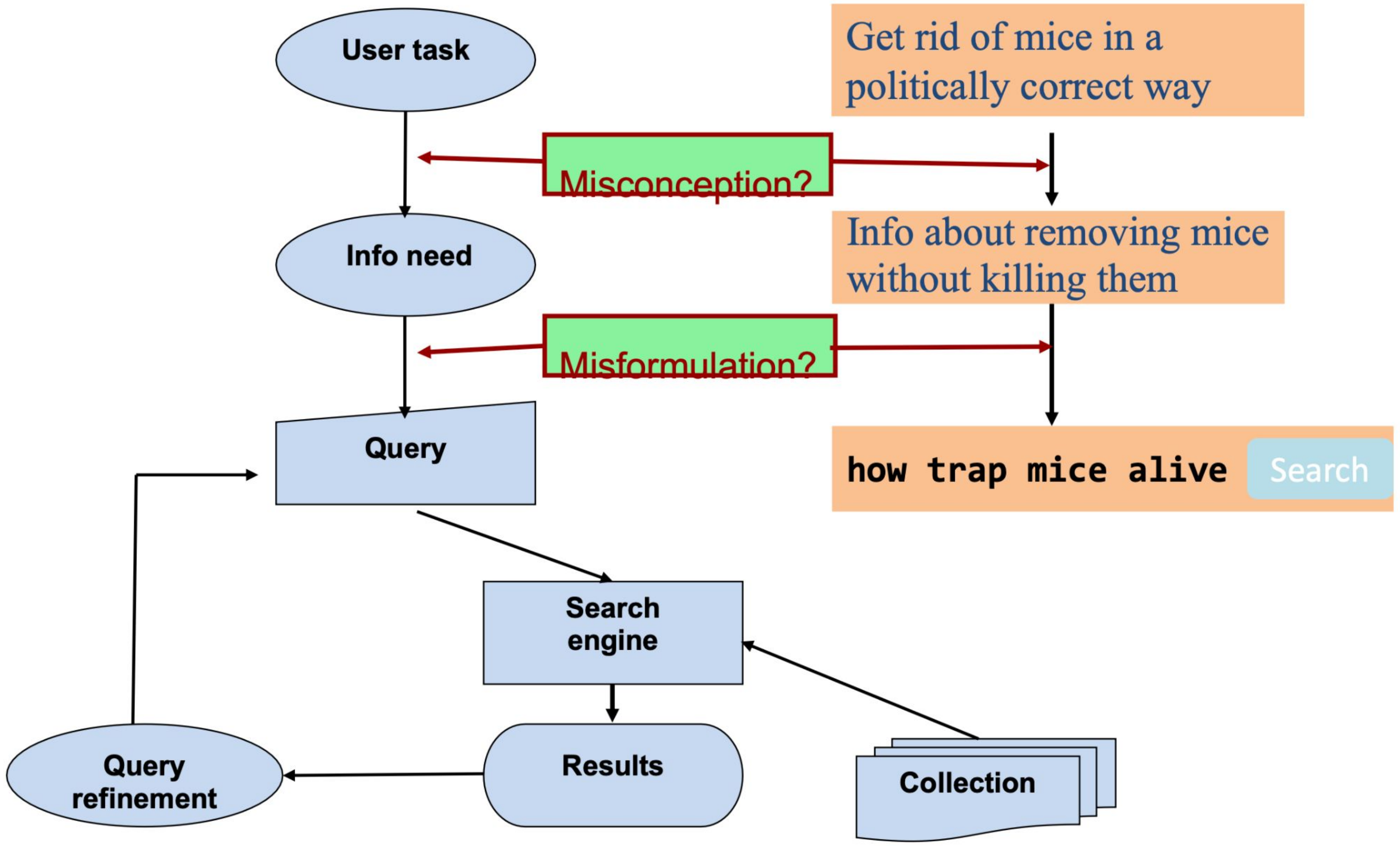


University at Buffalo
Department of CSE

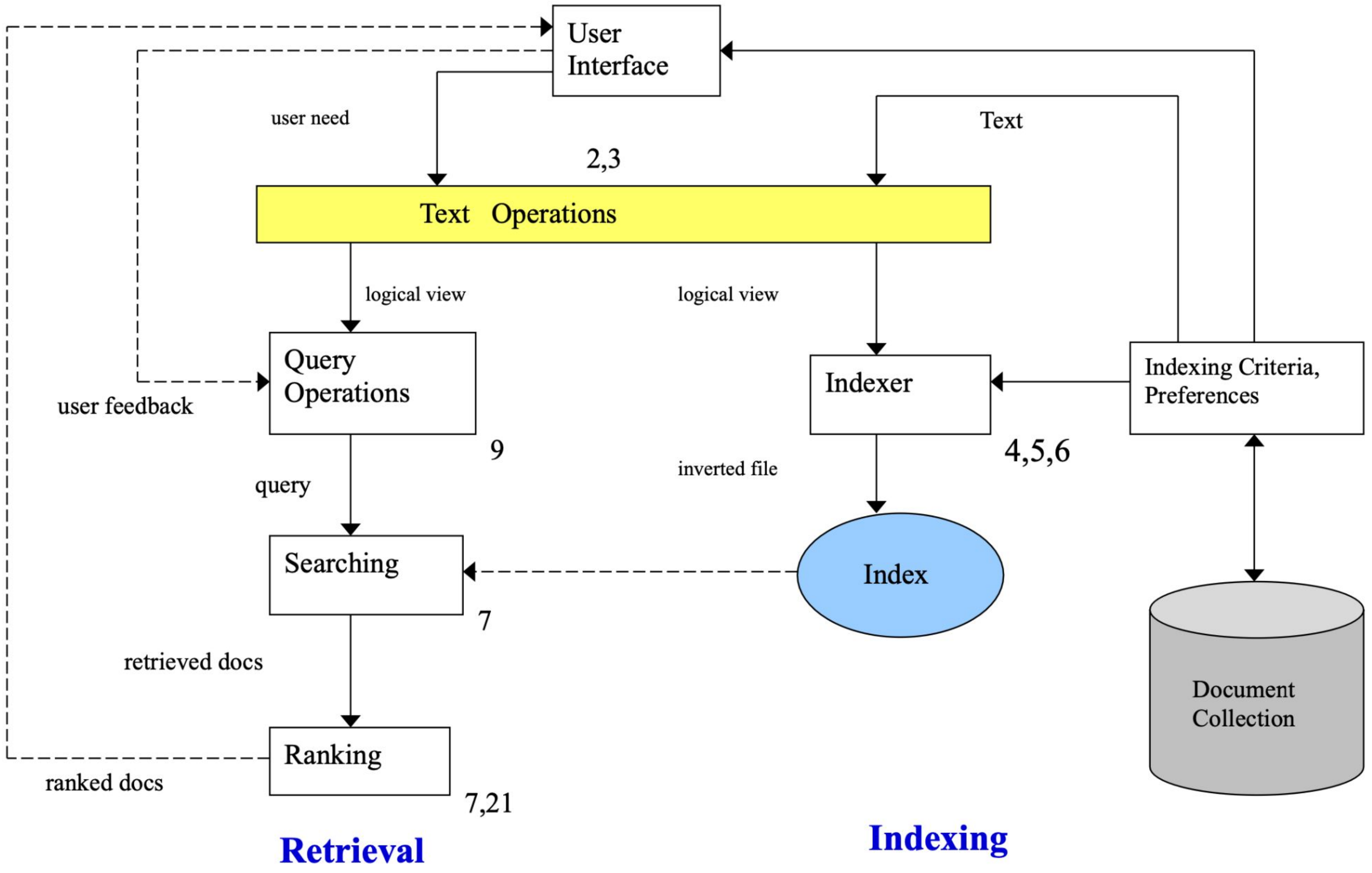
Basic assumptions of Information Retrieval

1. **Collection**: A set of documents
    a. Assume it is a static collection for the moment
2. **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



The Retrieval Process



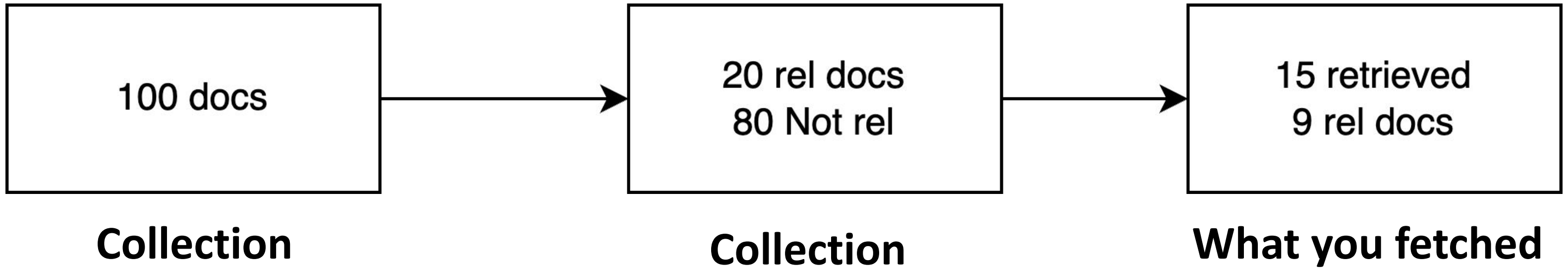
How good are the retrieved docs?

How good are the retrieved docs?

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

How good are the retrieved docs?

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



Precision : 9/15, Recall: 9/20

When Precision is Crucial: The Dating App Scenario

Imagine you're on a dating app and you've set some very specific filters:

- Loves dogs
- is a vegan,
- has a PhD in Astrophysics
- and must know the difference between "your" and "you're".

You only get three matches, but hey, they're perfect! You'd rather have these three ideal candidates than a hundred who only fulfill one of your criteria. This is like wanting high precision in your search engine; you want whatever comes up to be a perfect match, even if you get fewer results.

"In the dating app of life, precision ensures you don't swipe right on a cat person who thinks 'Star Wars' is a documentary."

When Recall is Important: The Easter Egg Hunt

Think of a classic Easter egg hunt; there are 100 eggs hidden, and some of them have money inside. You have only 5 minutes to find as many as possible. You're not too fussy about which eggs you're picking up; you want to collect as many as you can, money or not! This is when recall is more important than precision. It doesn't matter if you pick up a few empty or less desirable eggs; what matters is that you didn't miss out on the ones with the money.

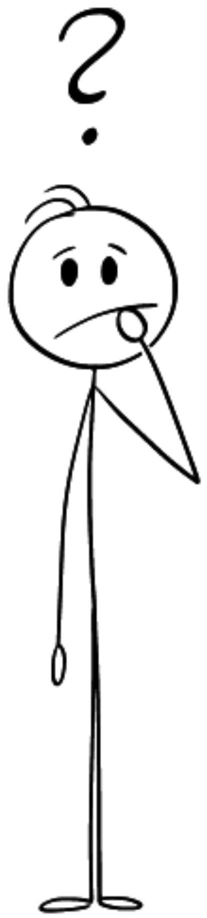
"When you're in the 'Easter egg hunt' mode, recall helps ensure you're not missing out on the golden eggs, even if you have to carry some empty shells along the way."

Unstructured data in 1620

- Which plays of Shakespeare contain the words Brutus AND Caesar but NOT Calpurnia?

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?



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)
  - Later lectures

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

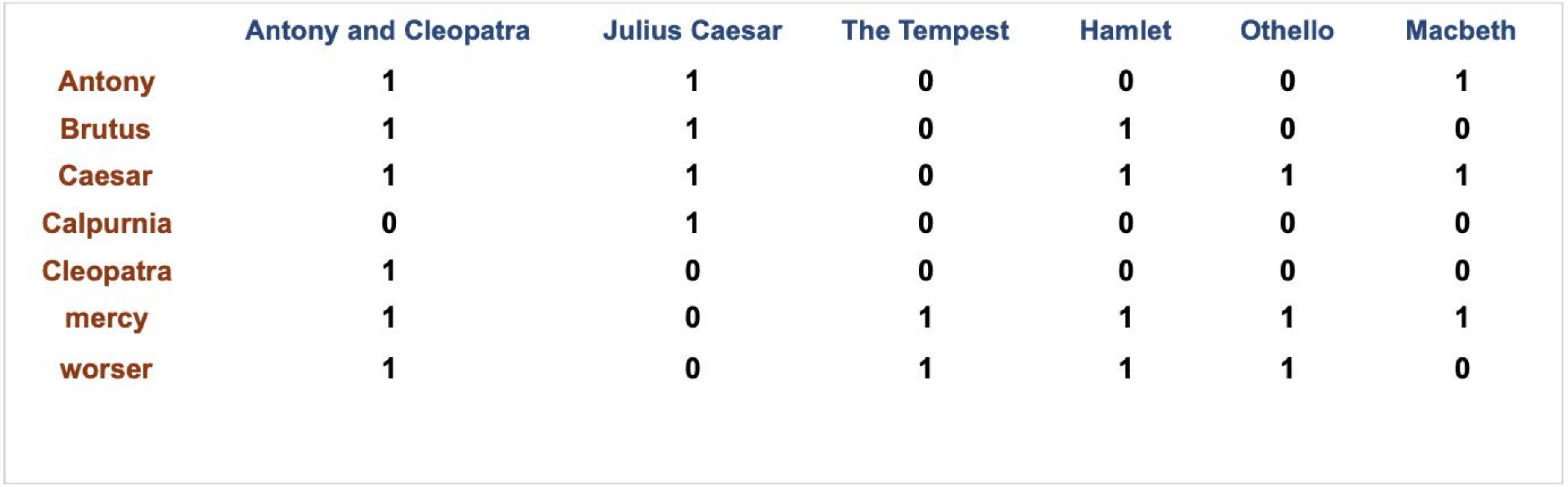
Brutus *AND* ***Caesar*** *BUT NOT* ***Calpurnia***

1 if play contains word, 0 otherwise

Incidence vectors

- So we have a 0/1 vector for each term.
- To answer query: take the vectors for Brutus, Caesar and 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,

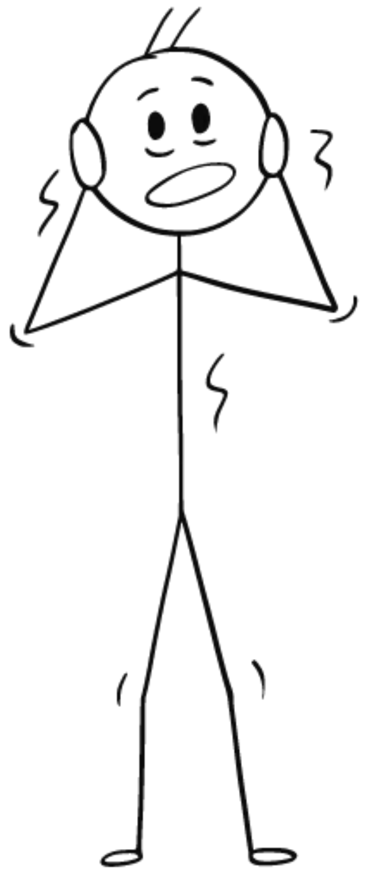
  He cried almost to roaring; and he wept When at Philippi he found **Brutus** slain.

- **Hamlet, Act III, Scene ii**

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

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.



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.
  - matrix is extremely sparse.

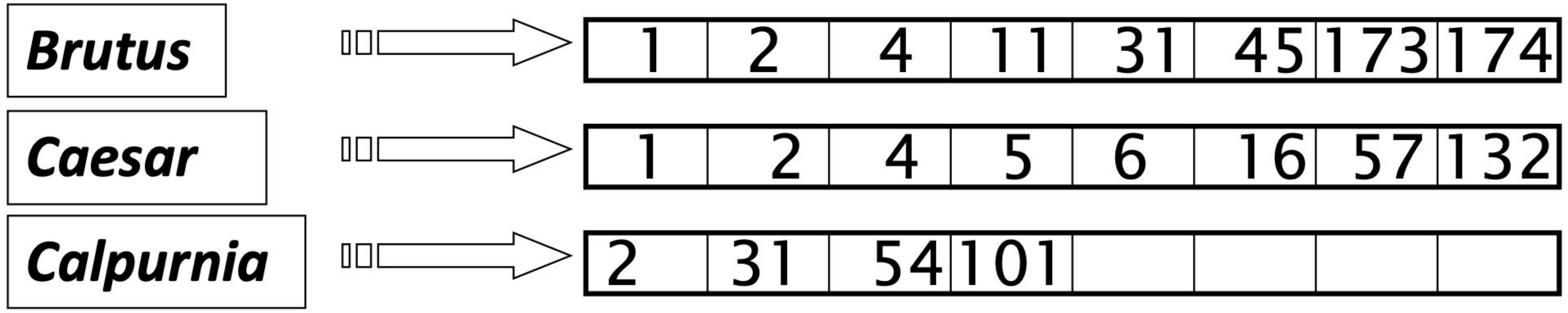
Can’t build the matrix

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

The Inverted Index
The key data structure underlying modern IR

Inverted index

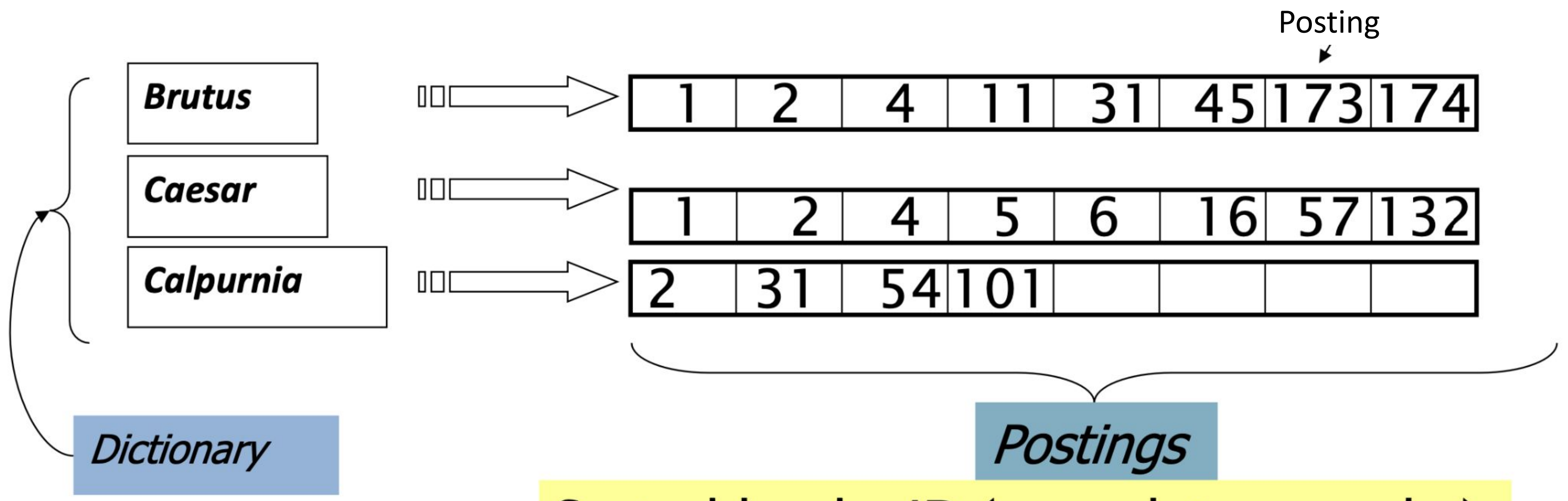
- For each term t, we must store a list of all documents that contain t.
  - Identify each doc by a docID, a document serial number
- Can we used fixed-size arrays for this?



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

Inverted index

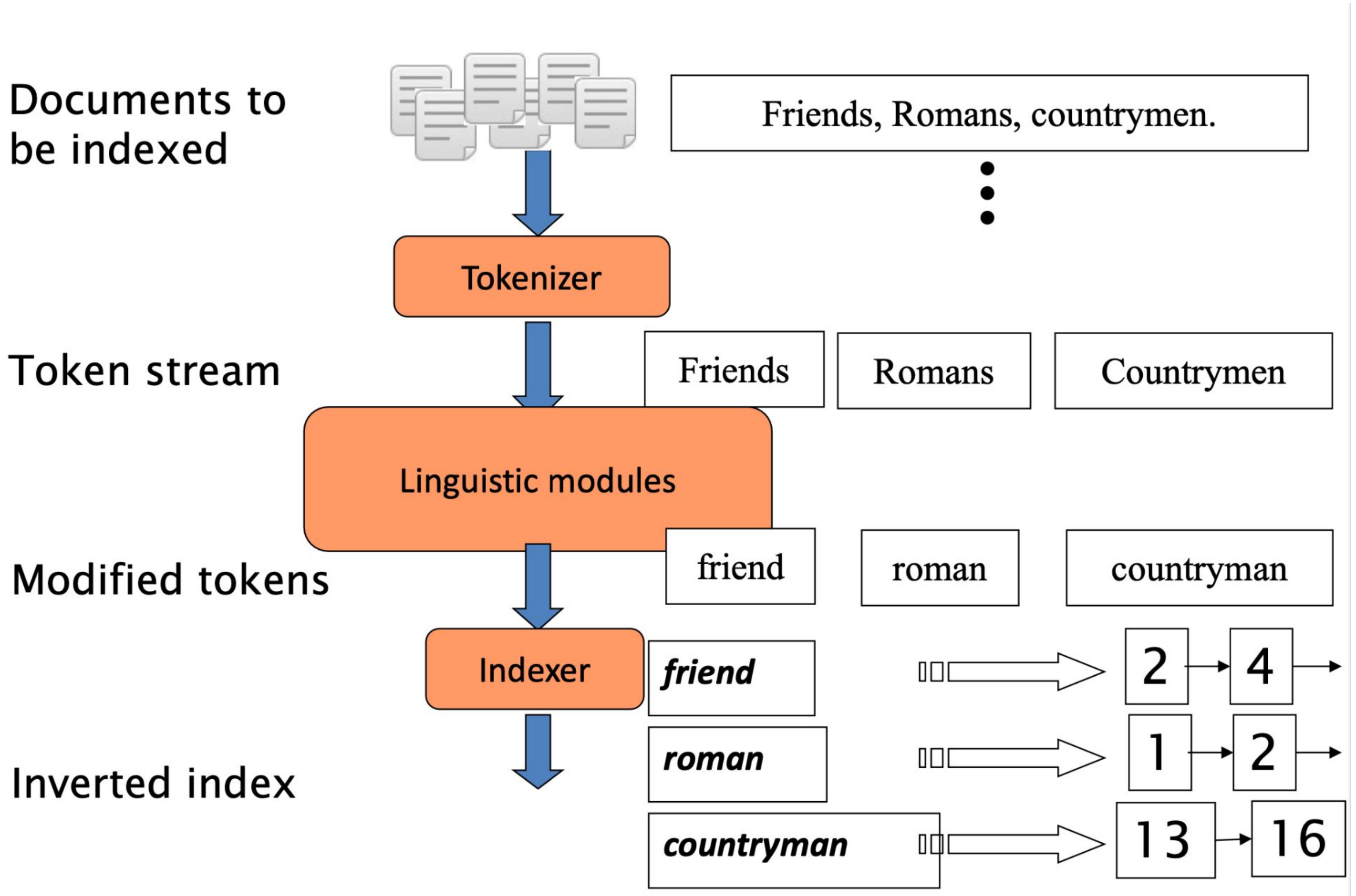
- We need variable-size postings lists
  - On disk, a continuous run of postings is normal and best
  - In memory, can use linked lists or variable length arrays
- Some tradeoffs in size/ease of insertion



Sorted by docID (more later on why).

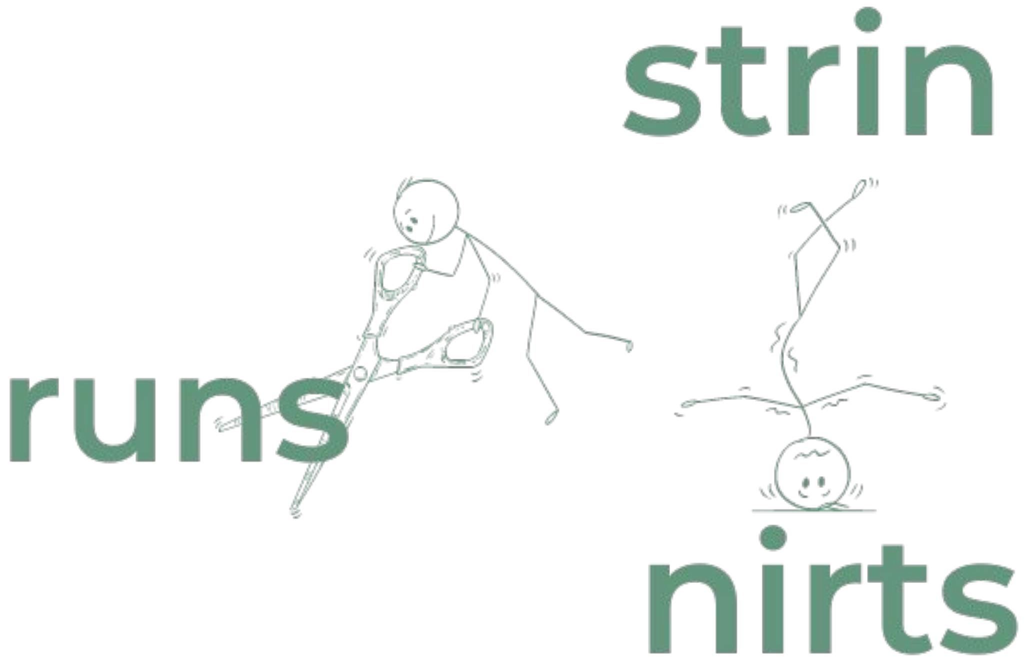
University at Buffalo
Department of CSE

Inverted Index construction



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/ Lemmatization
  - 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

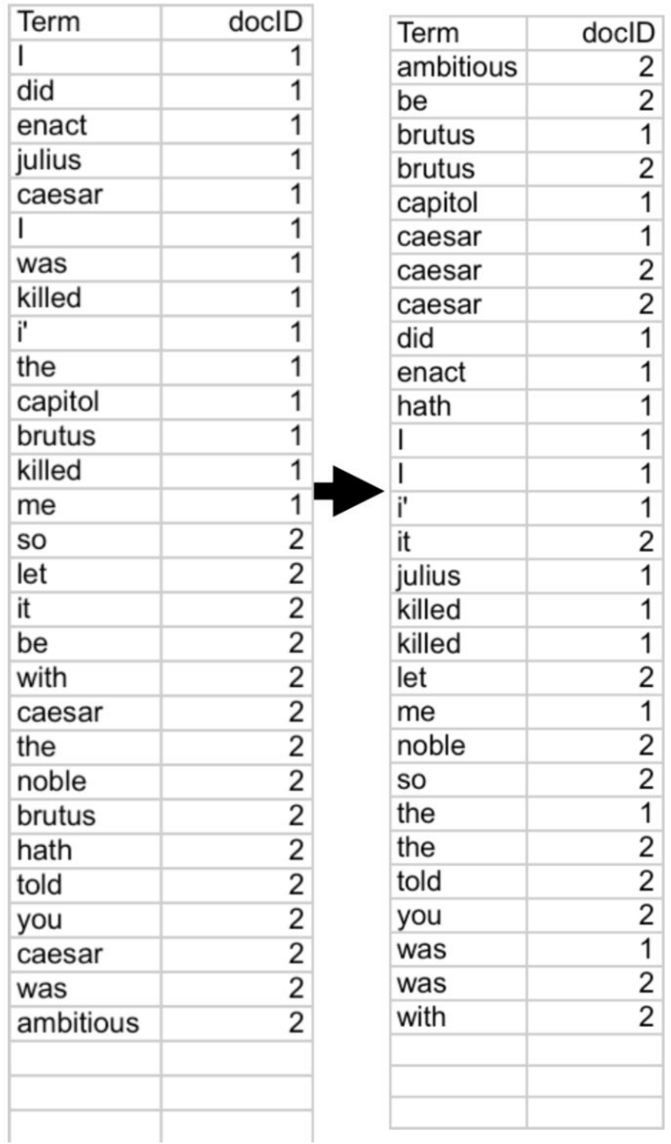
Term	docID
I	1
did	1
enact	1
julius	1
caesar	1
I	1
was	1
killed	1
i'	1
the	1
capitol	1
brutus	1
killed	1
me	1
so	2
let	2
it	2
be	2
with	2
caesar	2
the	2
noble	2
brutus	2
hath	2
told	2
you	2
caesar	2
was	2
ambitious	2

University at Buffalo
Department of CSE

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

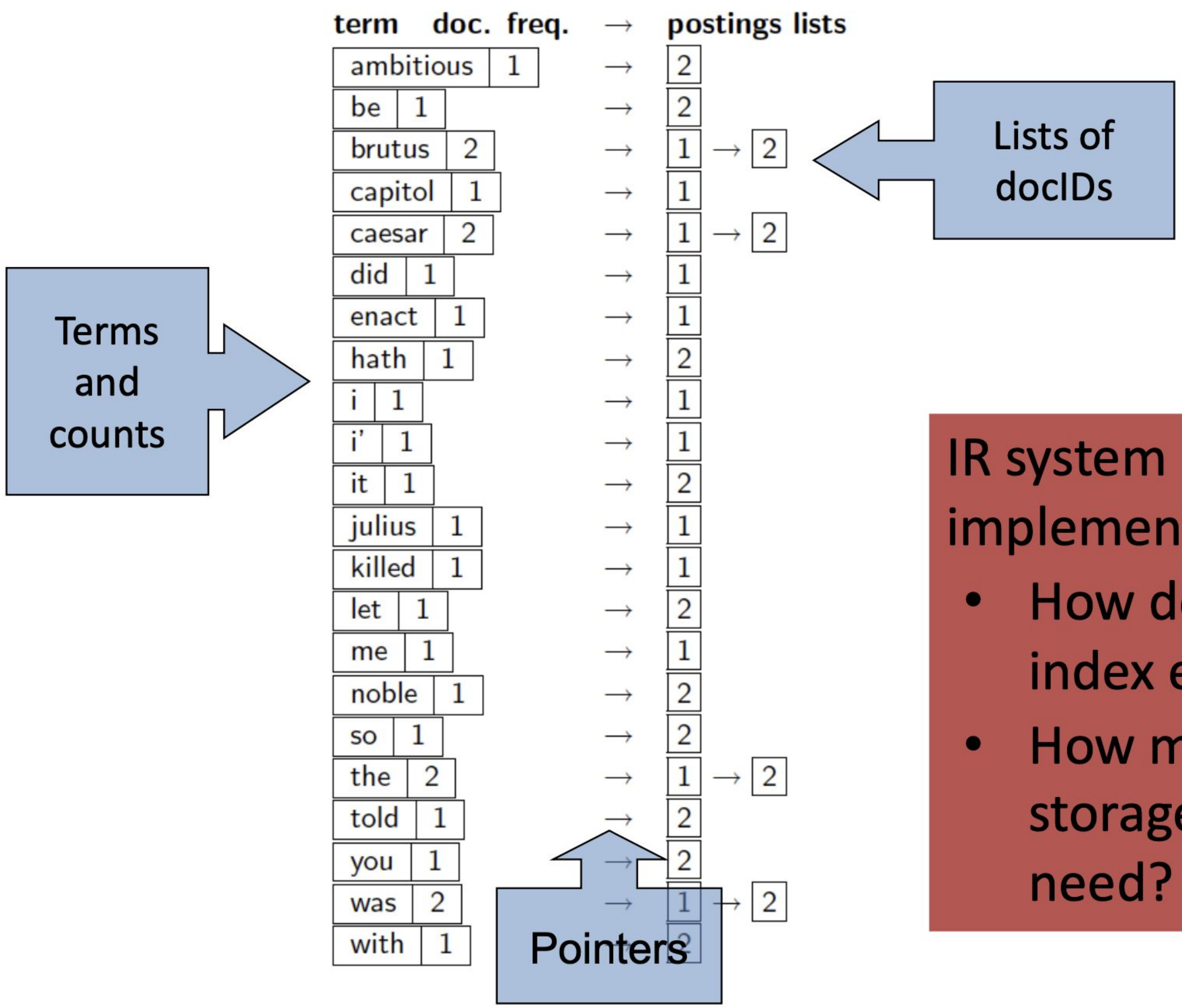
Indexer steps: Dictionary & Postings

- 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

Where do we pay in 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

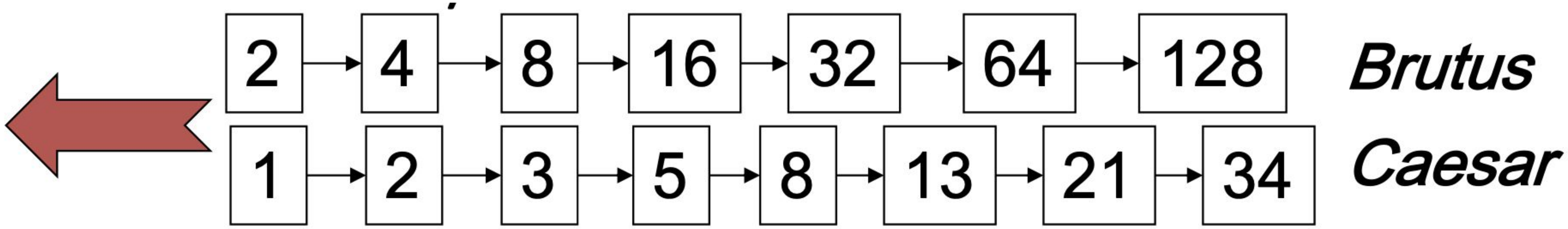
IR system implementation

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

Query processing with an inverted index

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):



Merge Algorithm

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

The Boolean Retrieval Model & Extended Boolean Models

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

Example: WestLaw

- 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?

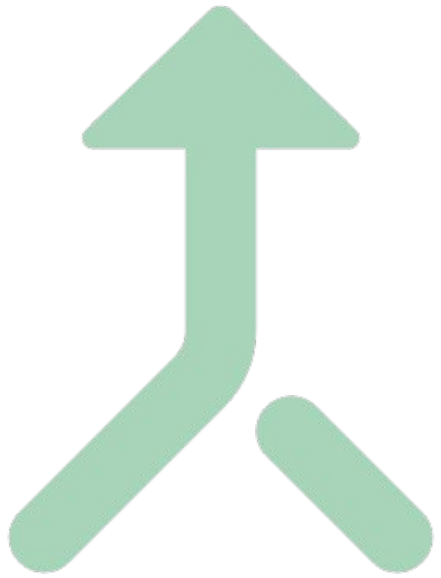
Merging

What about an arbitrary Boolean formula?

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

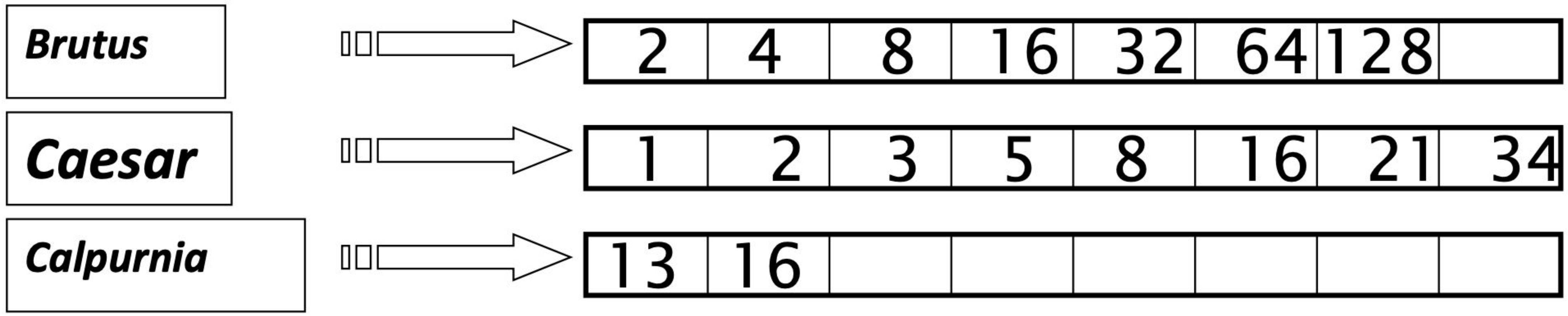
(**Antony** OR **Cleopatra**)

- Can we always merge in “linear” time?
  - Linear in what? (# of query terms, documents)
- Can we do better?



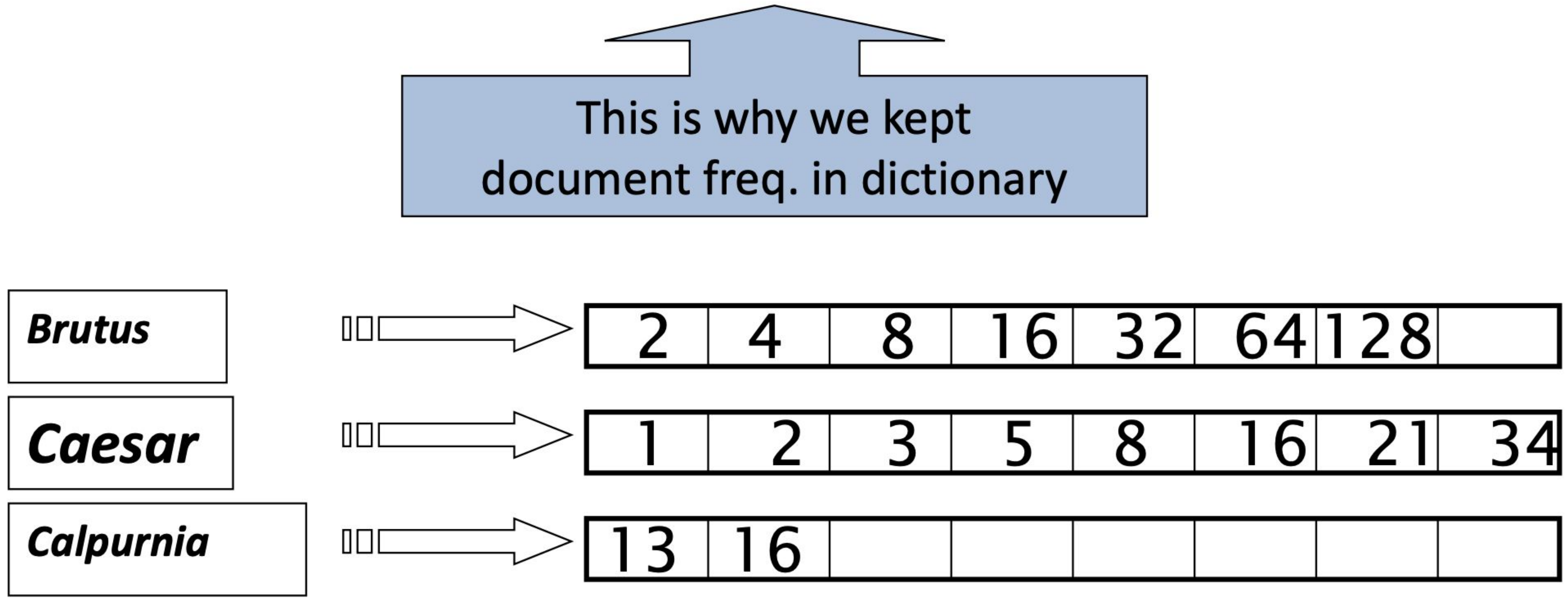
Query optimization

- 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.



Query optimization example

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



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

More general optimization

- E.g., (madding OR crowd) AND (ignoble OR strife)
- Conjunctive normal form (CNF)
- 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.

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

References

1. Slides provided by Sougata Saha (Instructor, Fall 2022 - CSE 4/535)
2. Materials provided by Dr. Rohini K Srihari
3. https://nlp.stanford.edu/IR-book/information-retrieval-book.html