COMP4650/6490 Document Analysis

Introduction to IR & Boolean Retrieval

ANU School of Computing

Administrative Matters

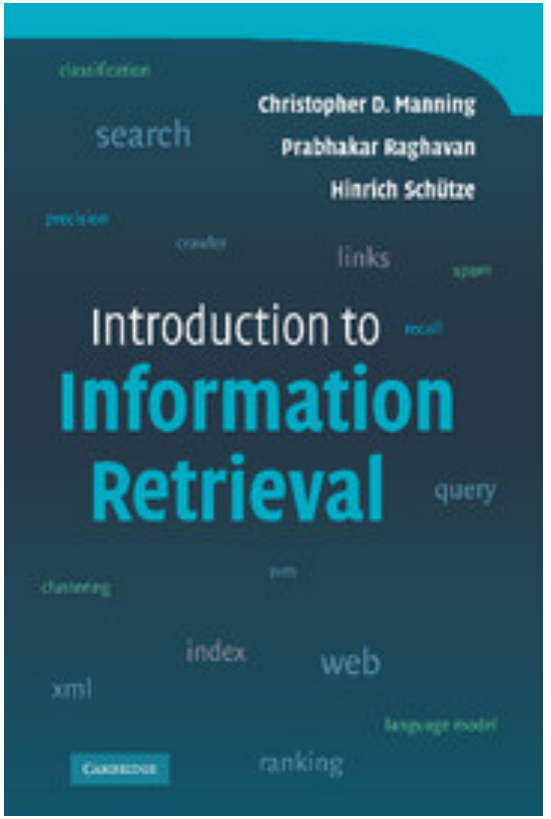
- COMP4650 representative nomination
  - email the convener by end of this week
- Lab 1
- Assignment 1
  - Release: 5pm Monday 31 July
  - Due: 5pm Wednesday 16 August, AEST (UTC +10)
  - Start working on it

IR Module Overview

- IR module consists of four lectures
  - Introduction to IR + Boolean model
  - Ranked retrieval model
  - Evaluation of IR systems
  - Web search basics

Textbook

- Introduction to information retrieval
- https://nlp.stanford.edu/IR-book/
- Chapters: 1, 2, 6, 8, 21
  (chapters 16 and 18 in ML module)



Australian National University

Outline

- Introduction to IR
  - What is information retrieval
  - Why information retrieval
  - How to perform information retrieval
  - IR vs NLP
- Boolean retrieval
  - Indexing, querying, and retrieval procedures
  - Term-document incidence matrix
  - Inverted index
  - Boolean retrieval with inverted index
- Initial stages of text processing

Australian National University

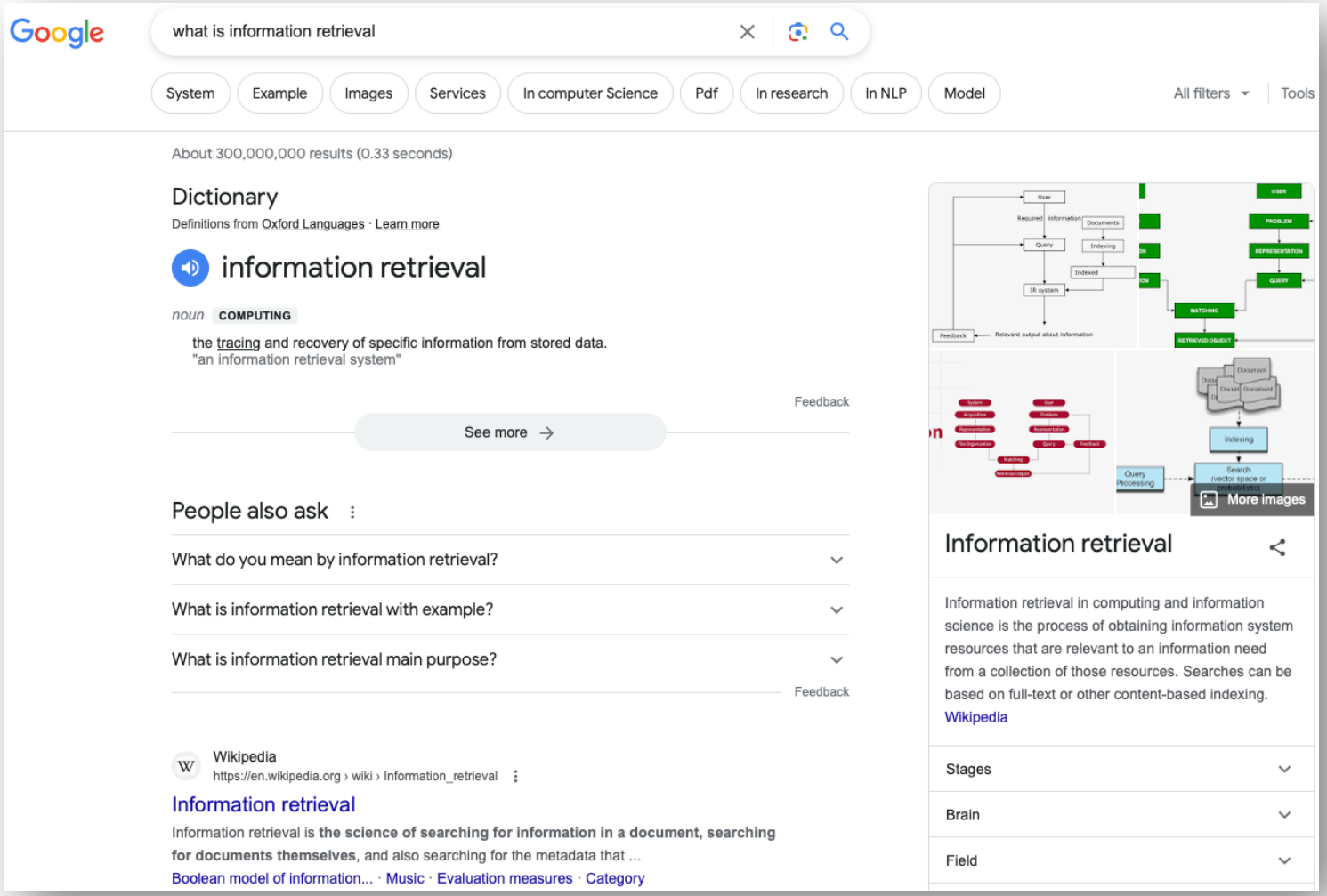
Outline

- Introduction to IR
  - What is information retrieval
  - Why information retrieval
  - How to perform information retrieval
  - IR vs NLP
- Boolean retrieval
  - Indexing, querying, and retrieval procedures
  - Term-document incidence matrix
  - Inverted index
  - Boolean retrieval with inverted index
- Initial stages of text processing

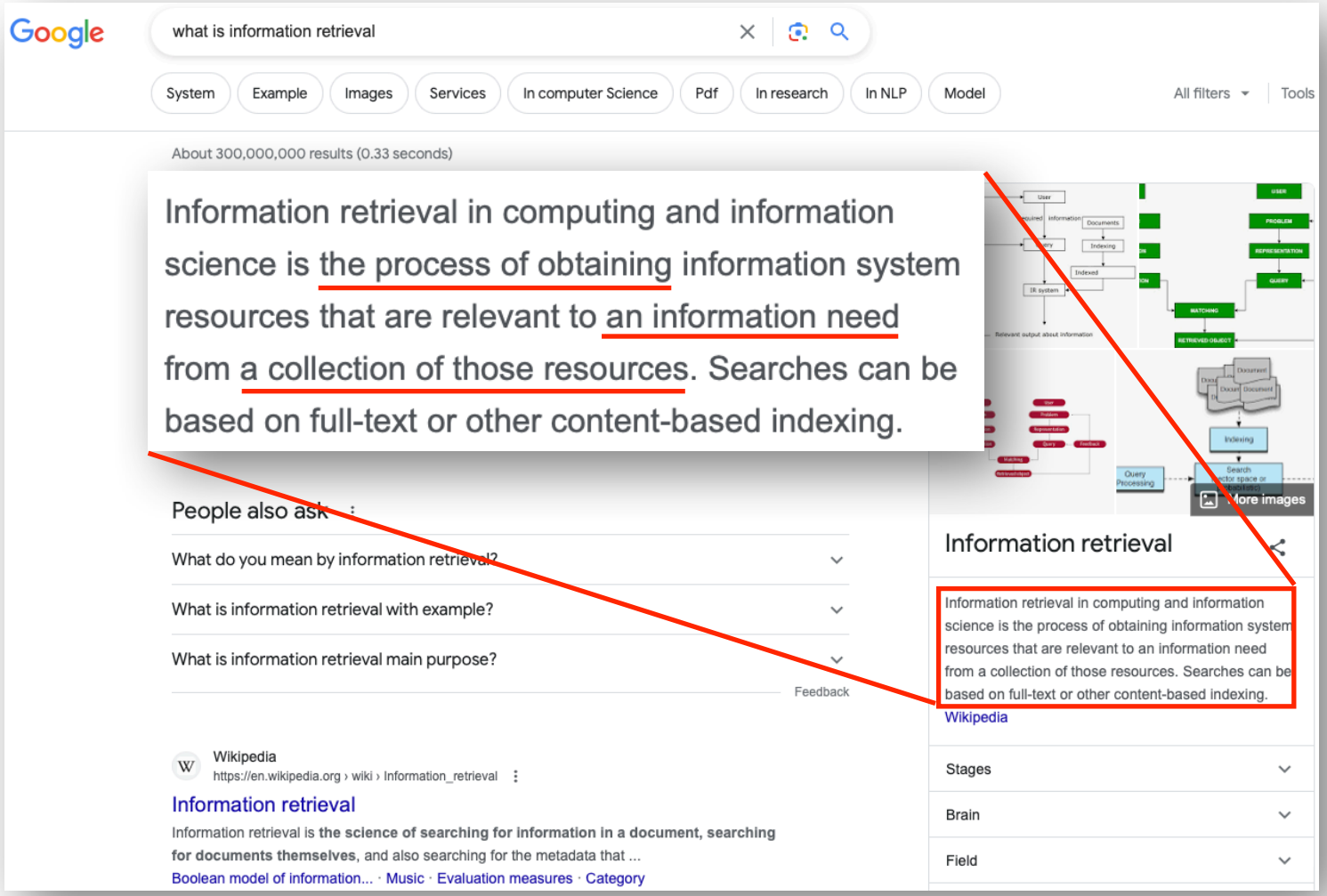
Introduction to IR

What is information retrieval

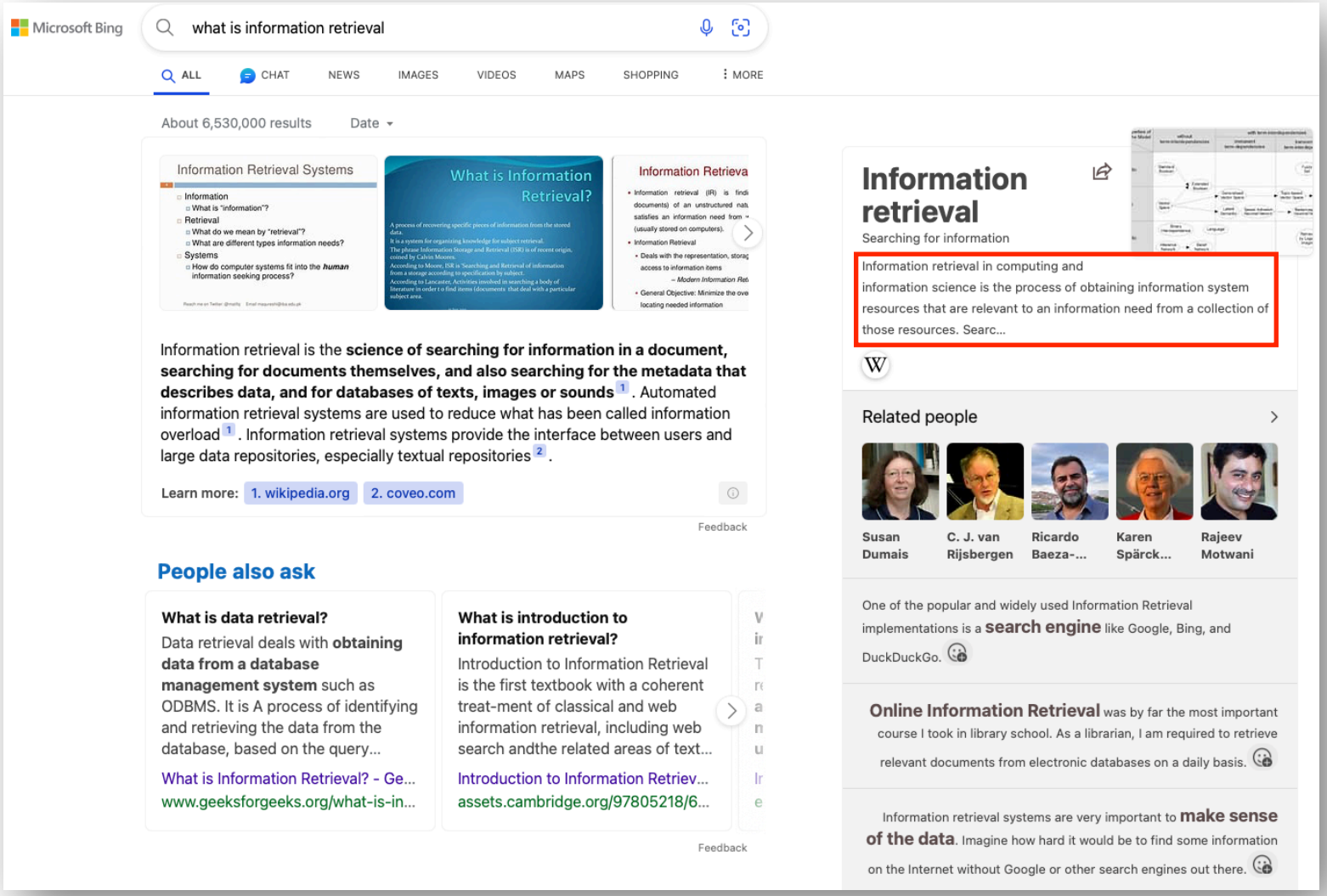
What is information retrieval



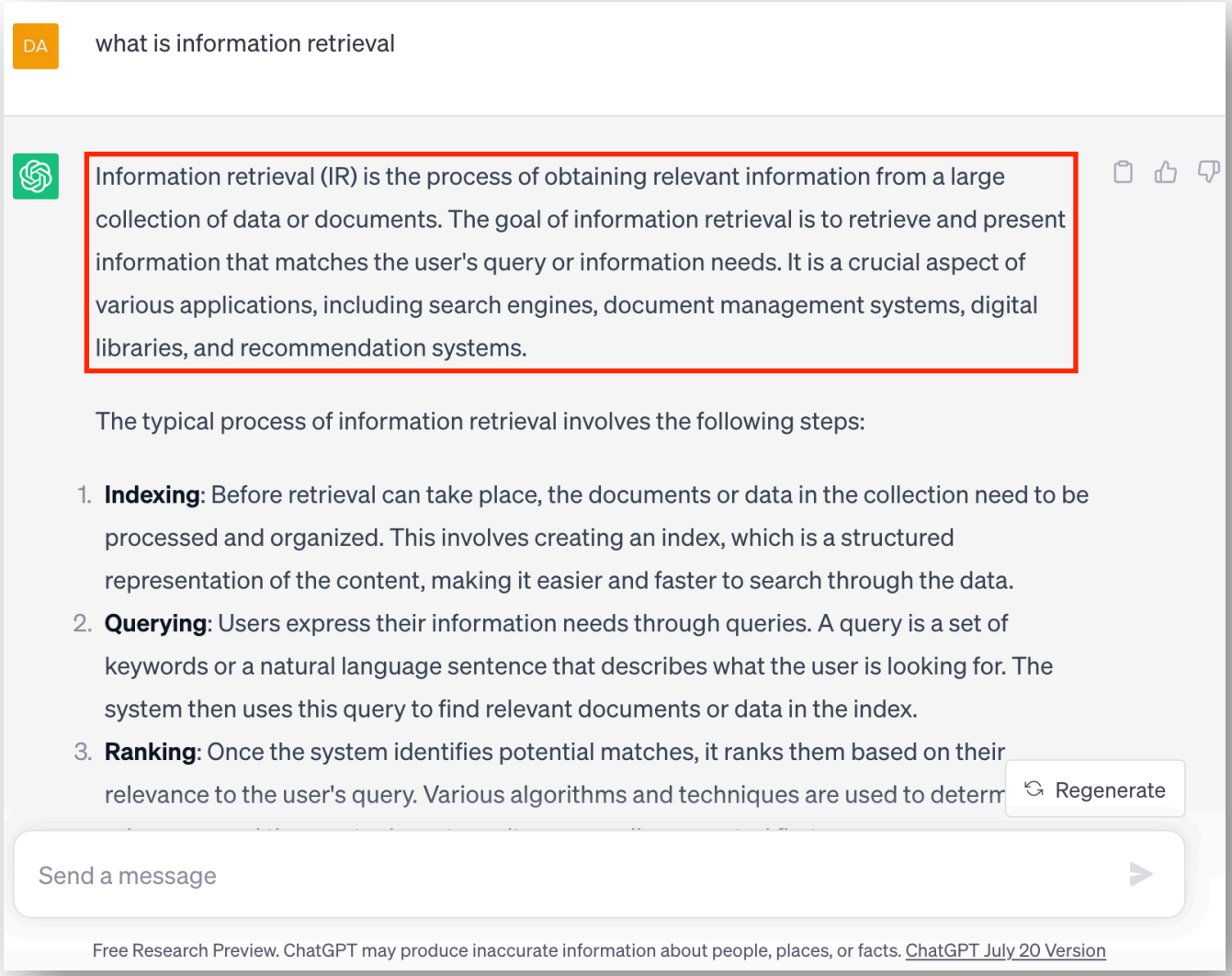
What is information retrieval



What is information retrieval



What is information retrieval



Introduction to IR

What is information retrieval

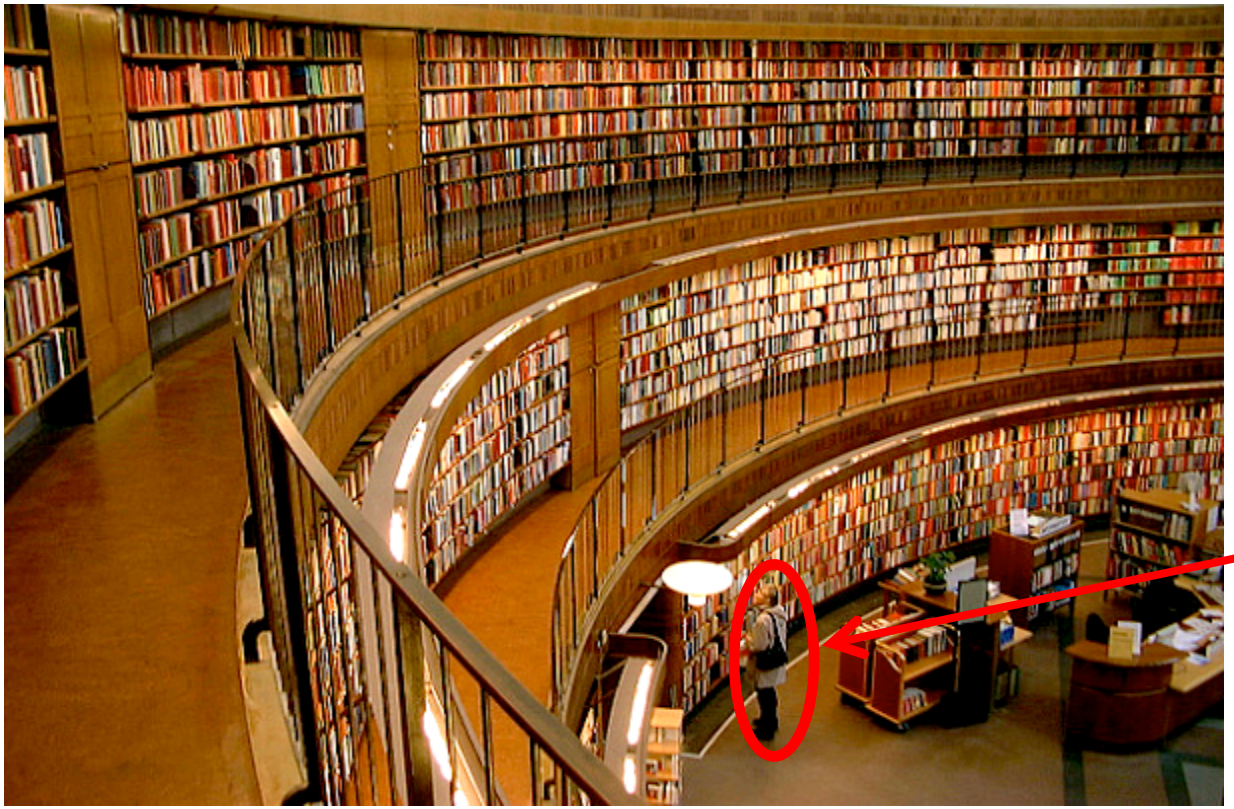
- *"Information Retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).*" — Manning et al.
- You may think of *web search* first, but there are many other cases
  - E-mail search
  - Searching your laptop
  - Corporate knowledge bases
  - Image search, video search

Why information retrieval

"*Information overload is the difficulty in understanding an issue and effectively making decisions when one has too much information about that issue, and is generally associated with the excessive quantity of daily information*." - Wikipedia

Why information retrieval

IR is an essential tool to deal with information overload



How to perform information retrieval

Information retrieval when we did not have a computer

How to perform information retrieval

Starting point

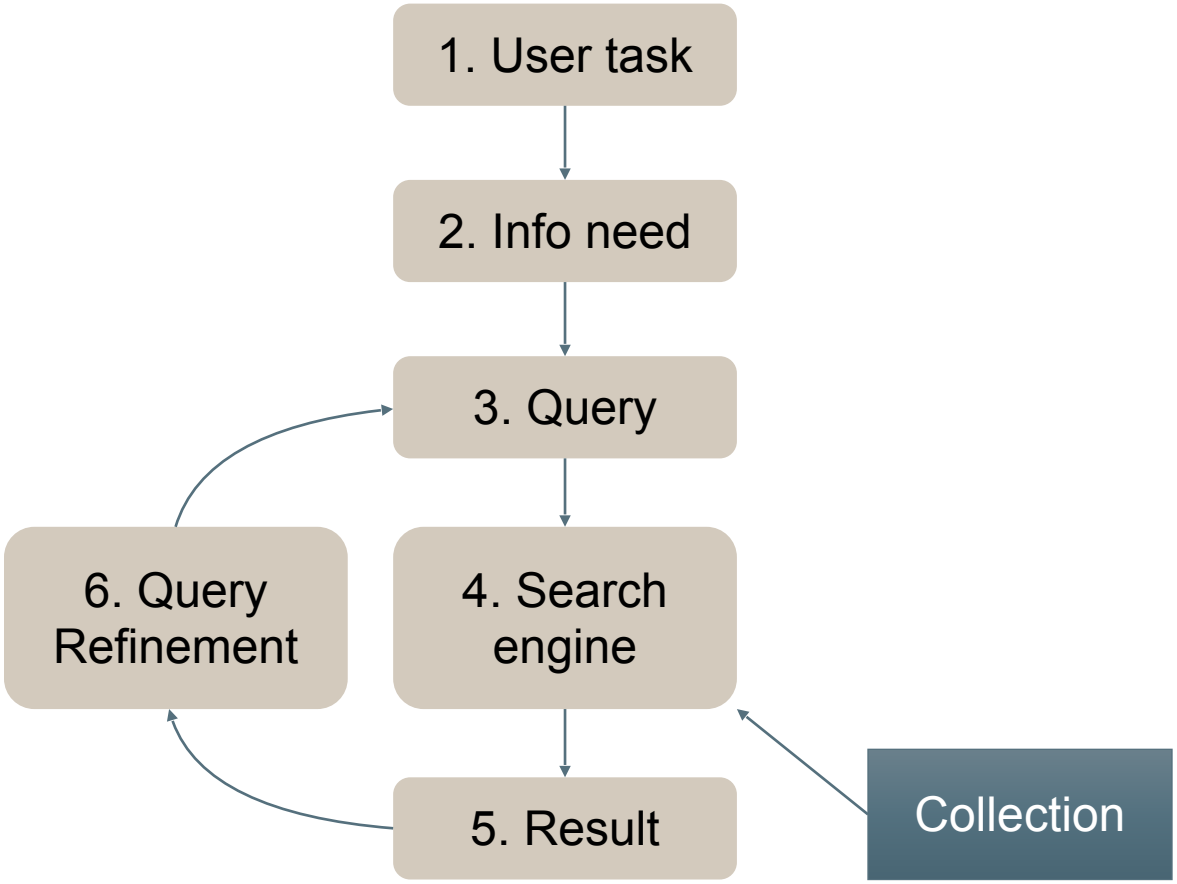
- Collection: a set of documents
  - Assume it as a static collection for the moment
- Goal: Retrieve documents with information that is relevant to the user's information need and helps the user complete a task
  - User's information need is often underspecified

Key objectives: Every good IR system needs to achieve

- Scalability: 40+ billion pages are indexed by Google
- Accuracy: Top-10 pages from 40+ billion pages?

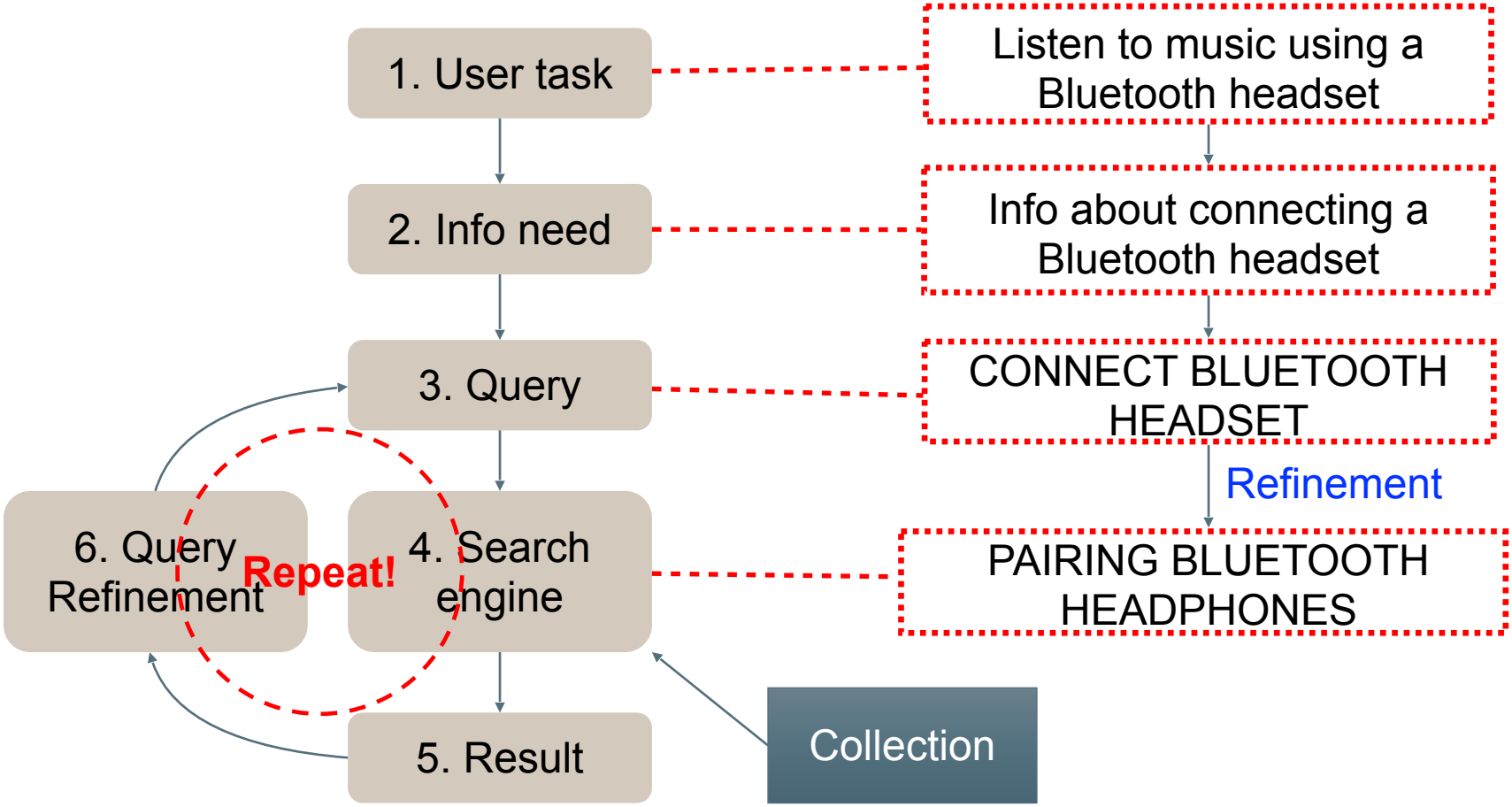
How to perform information retrieval

Classic search model



How to perform information retrieval

Classic search model



IR vs NLP

- Information retrieval
  - Computational approaches
  - Statistical (shallow) understanding of language
  - Handle large scale problems
- Natural language processing
  - Cognitive, symbolic and computational approaches
  - Semantic (deep) understanding of language
  - (often) smaller scale problems

Introduction to IR

IR vs NLP

- IR and NLP are getting closer
- IR for NLP
  - Larger data collections
  - Scalable / robust NLP techniques
    e.g. translation models
- NLP for IR
  - Deep analysis of text documents and queries
  - Information extraction for structured IR tasks
  - Conversational IR

Australian National University

Outline

- Introduction to IR
  - What is information retrieval
  - Why information retrieval
  - How to perform information retrieval
  - IR vs NLP
- Boolean retrieval
  - Indexing, querying, and retrieval procedures
  - Term-document incidence matrix
  - Inverted index
  - Boolean retrieval with inverted index
- Initial stages of text processing

Boolean Retrieval

Indexing, Querying & Retrieval Procedures

- Indexing
  - Storing a mapping from terms to documents
- Querying
  - Looking up terms in the index and returning documents
  - Boolean query: e.g. “canberra” AND “healthcare” NOT “covid”
- Boolean retrieval procedures
  - Lookup query term in the dictionary
  - Retrieve the list of relevant documents (i.e. posting lists)
  - Operations
    - AND: intersect the posting lists
    - OR: union the posting lists
    - NOT: diff the posting lists

Boolean Retrieval

Indexing, Querying & Retrieval Procedures

- Modern IR retrieval procedures
  - Boolean model provides *all* the ranking candidates
    - Locate documents satisfying Boolean condition
      e.g. “travel insurance” => “travel” AND “insurance”
  - Rank candidates by relevance
    - Important: the notion of relevance
  - Efficiency consideration
    - Top-k retrieval (e.g. Google)

Boolean Retrieval

Term-Document Incidence 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

1 if play contains word, 0 otherwise

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

Boolean Retrieval

Term-Document Incidence Matrix

Words / Terms

Shakespeare Plays

	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

Boolean Retrieval

Term-Document Incidence Matrix

- Incidence vectors
  - We have a 0/1 vector for each term
- To answer the query Brutus AND Caesar BUT NOT Calpurnia
  - Take the vectors for *Brutus*, *Caesar* and *Calpurnia* (complemented)
  - Perform 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

Boolean Retrieval

Term-Document Incidence Matrix

Efficiency

- Bigger collections
  - 1 million documents
  - Each 1,000 words long
- Avg 6 bytes per word including spaces/punctuation
  - 6GB of data in the documents
- Assume there are 500K distinct terms among these
  - Corresponds to a matrix with 500 billion entries
  - But it has no more than one billion 1's
  - Extremely sparse matrix!

Boolean Retrieval

Term-Document Incidence Matrix

Efficiency

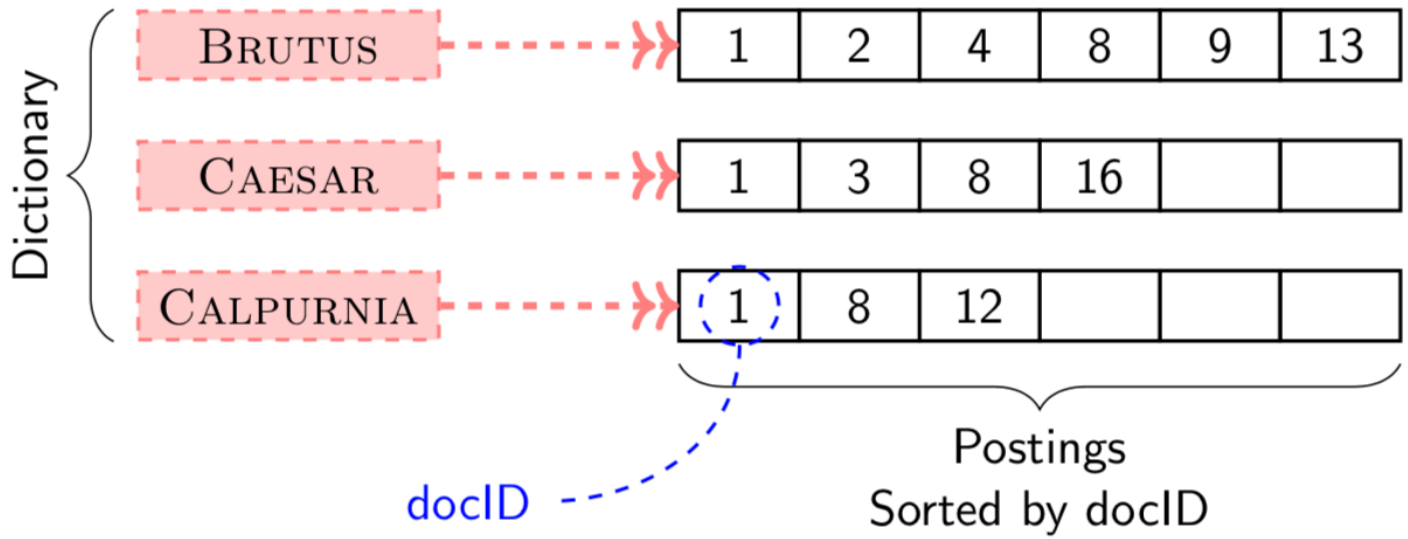
- Bigger collections
  - 1 million documents
  - Each 1,000 words long
- Avg 6 bytes per word including spaces/punctuation
  - 6GB of data in the documents
- Assume there are 500K distinct terms among these
  - Corresponds to a matrix with 500 billion entries
  - But it has no more than one billion 1's
  - Extremely sparse matrix!

Efficient data structure
tailored for boolean retrieval
=> Inverted index!

Boolean Retrieval

Inverted Index

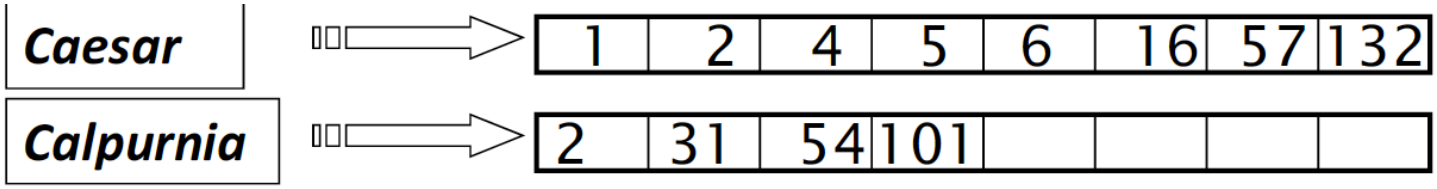
- Inverted index consists of a dictionary and postings
  - *Dictionary* (key): a set of unique terms
  - *Posting* (value): variable-size array that keeps the list of documents given term (sorted)
- Indexer: Construct inverted index from raw text



Boolean Retrieval

Inverted Index

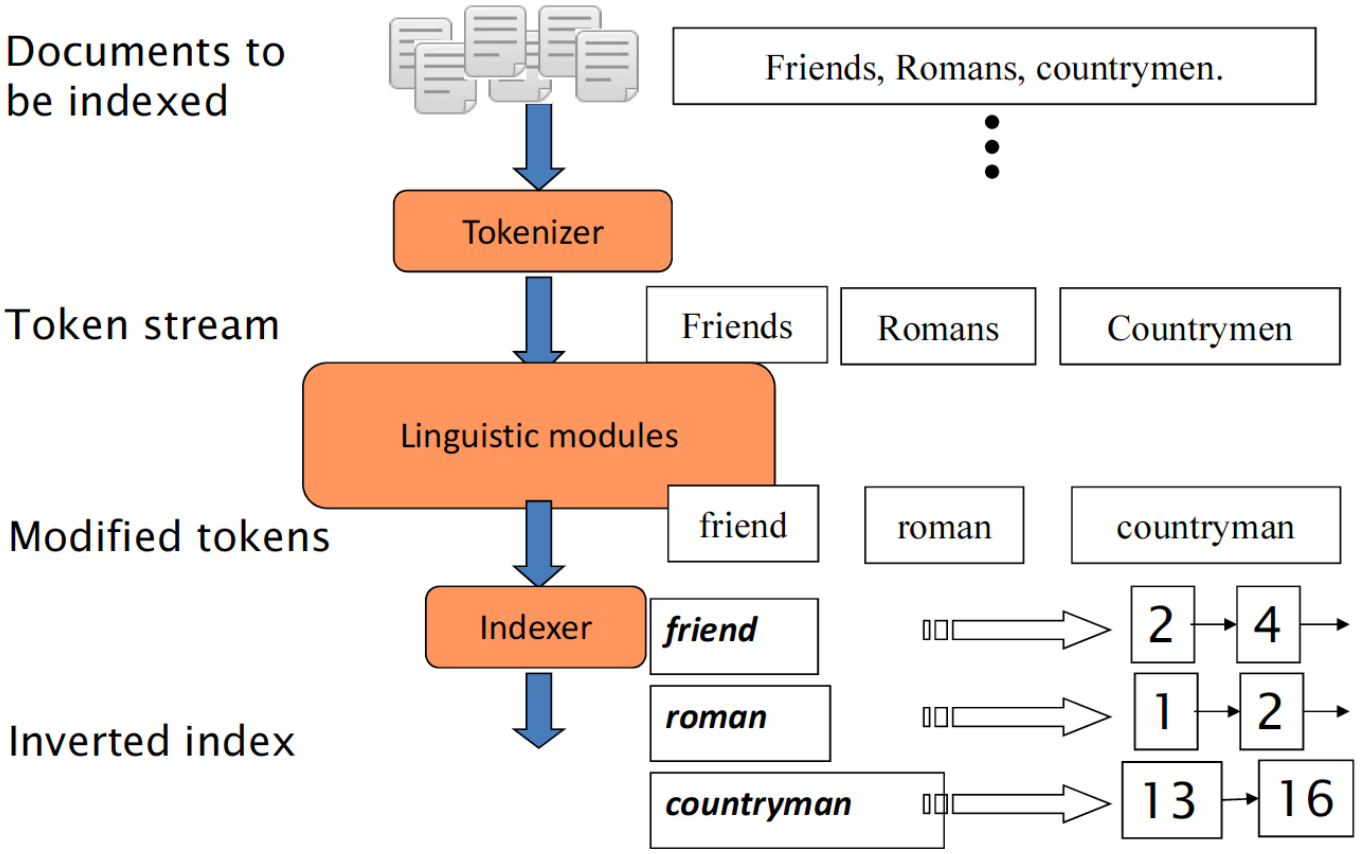
- For each term *t*, we must store a list of all documents that contain *t*
  - Identify each doc by a *docID*, a document serial number
- We need variable-size *postings lists*
  - On disk, a contiguous run of postings is normal and best
  - In memory, can use linked lists or variable length arrays



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

Boolean Retrieval

Inverted Index



Boolean Retrieval

Inverted Index

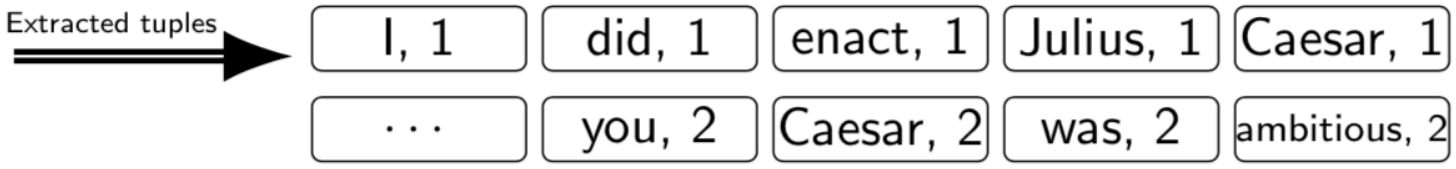
- Indexer Step 1: Token sequence
  - Scan documents for indexable terms
  - Keep list of (token, docID) pairs

doc 1

I did enact Julius Caesar: I was killed in the Capitol; Brutus killed me.

doc 2

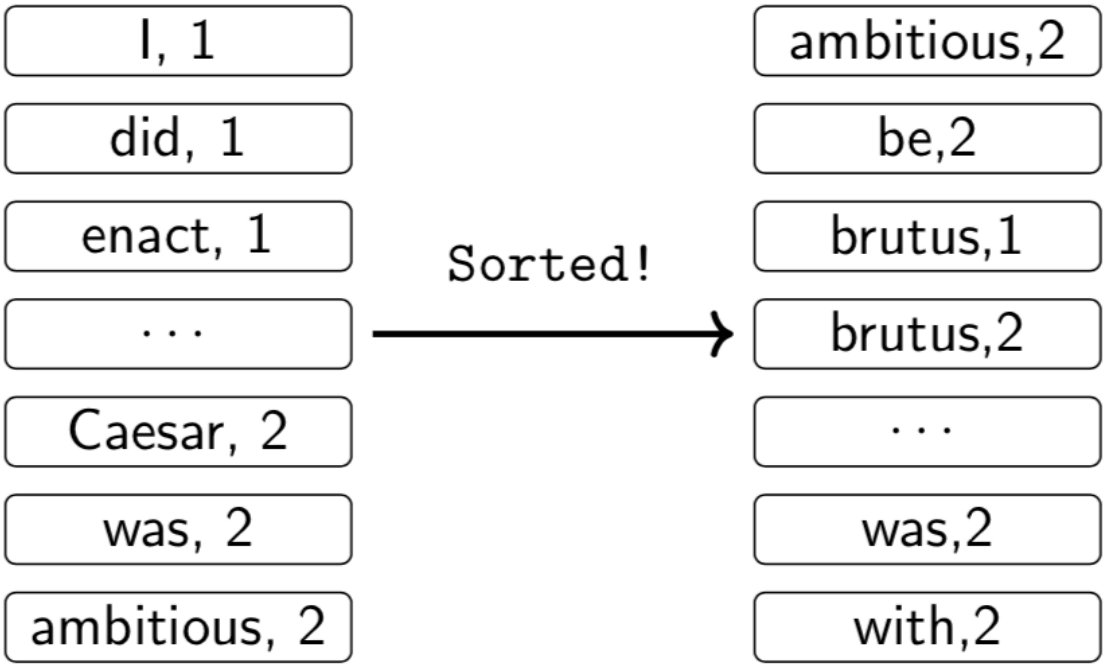
So let it be with Caesar. The noble Brutus hath told you Caesar was ambitious.



Boolean Retrieval

Inverted Index

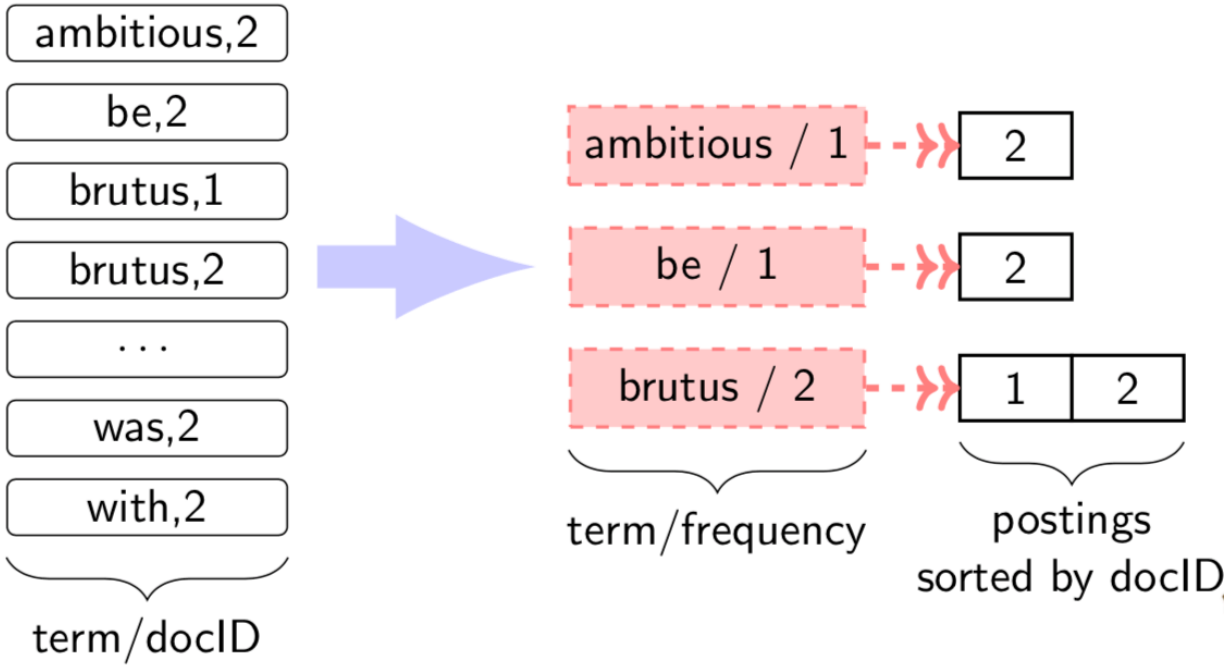
- Indexer Step 2
  - Sort tuples by *terms* (and then *docID*)



Boolean Retrieval

Inverted Index

- Indexer Step 3
  - Multiple term entries in a single document are merged
  - Split into *Dictionary* and *Postings*
  - Doc frequency information is added



Boolean Retrieval

Boolean Retrieval with Inverted Index

- Easy to retrieve all documents containing a term
- How to find documents containing *Brutus* AND *Caesar* using postings?
  - Linear time intersection algorithm for sorted lists

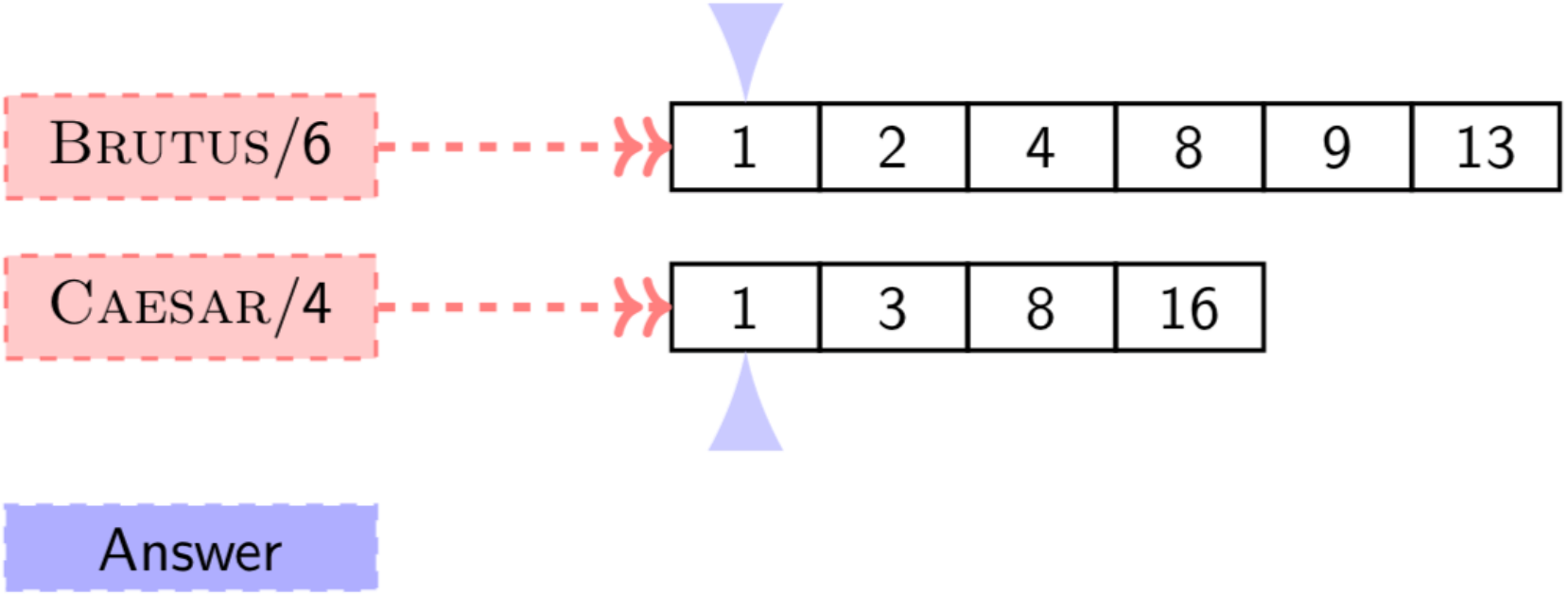
Algorithm 1 Intersection(p_1, p_2)

```
1: answer ← {}
2: while p1 ≠ NIL and p2 ≠ NIL do
3:     if docID(p1) = docID(p2) then
4:         ADD(answer, docID(p1)); p1 ← next(p1); p2 ← next(p2)
5:     else if docID(p1) < docID(p2) then
6:         p1 ← next(p1)
7:     else
8:         p2 ← next(p2)
9:     end if
10: end while
```

Boolean Retrieval

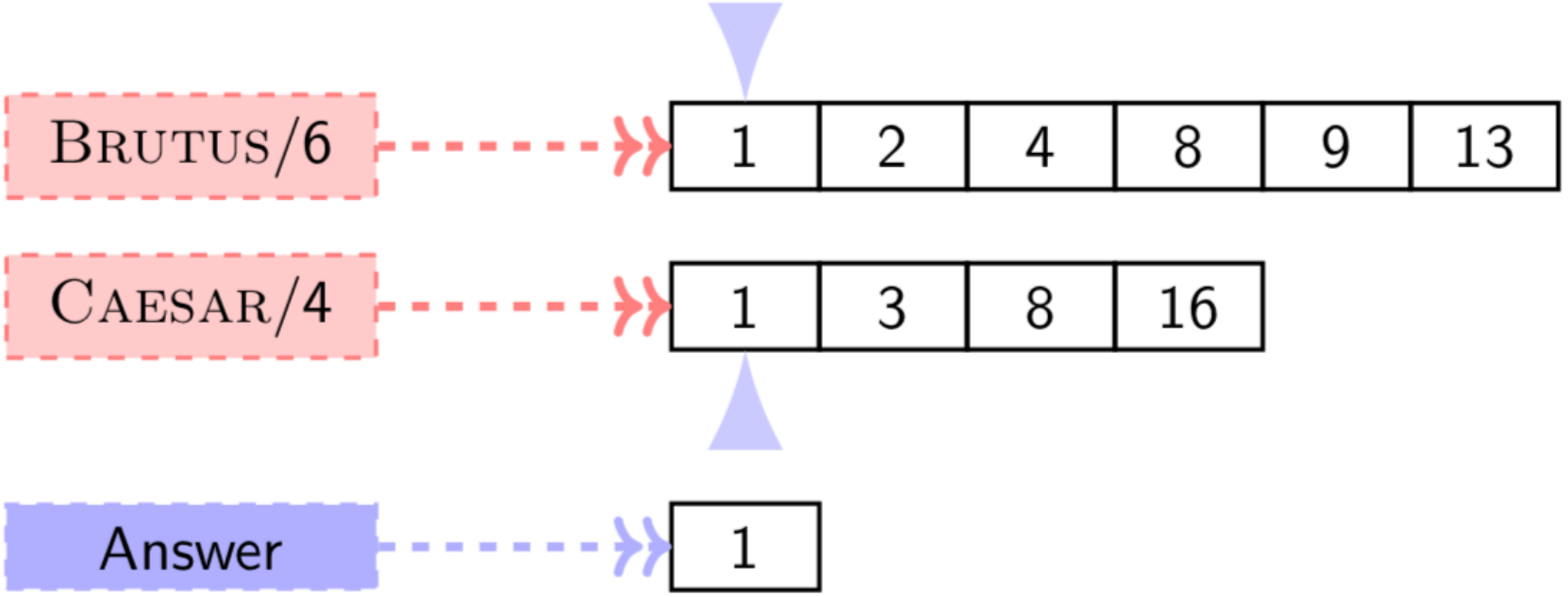
Boolean Retrieval with Inverted Index

- BRUTUS AND CEASAR



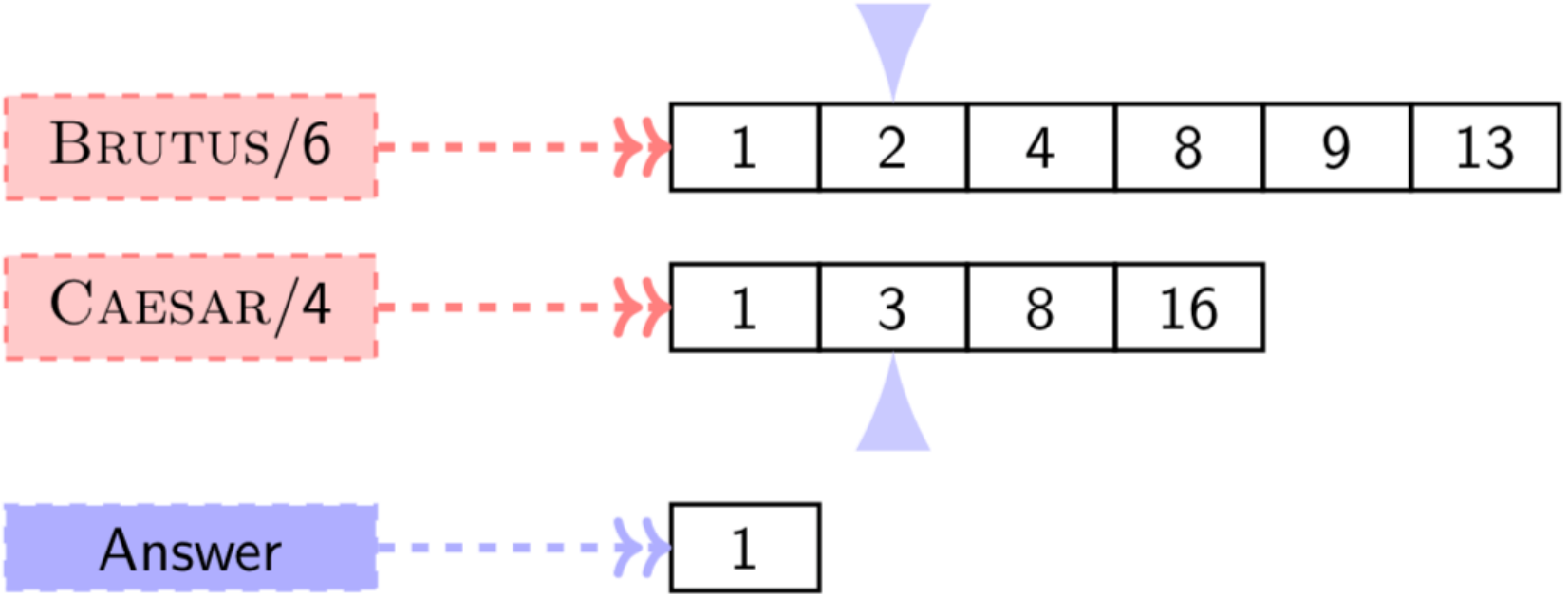
Boolean Retrieval

Boolean Retrieval with Inverted Index



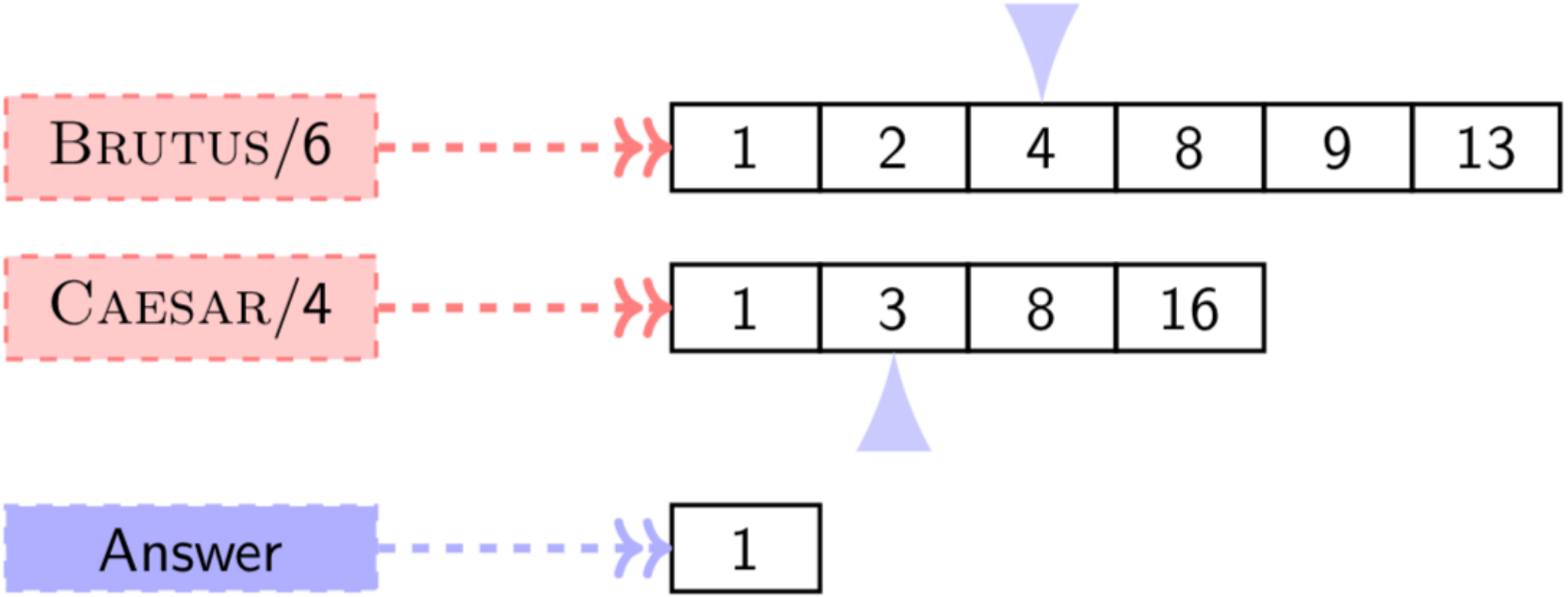
Boolean Retrieval

Boolean Retrieval with Inverted Index



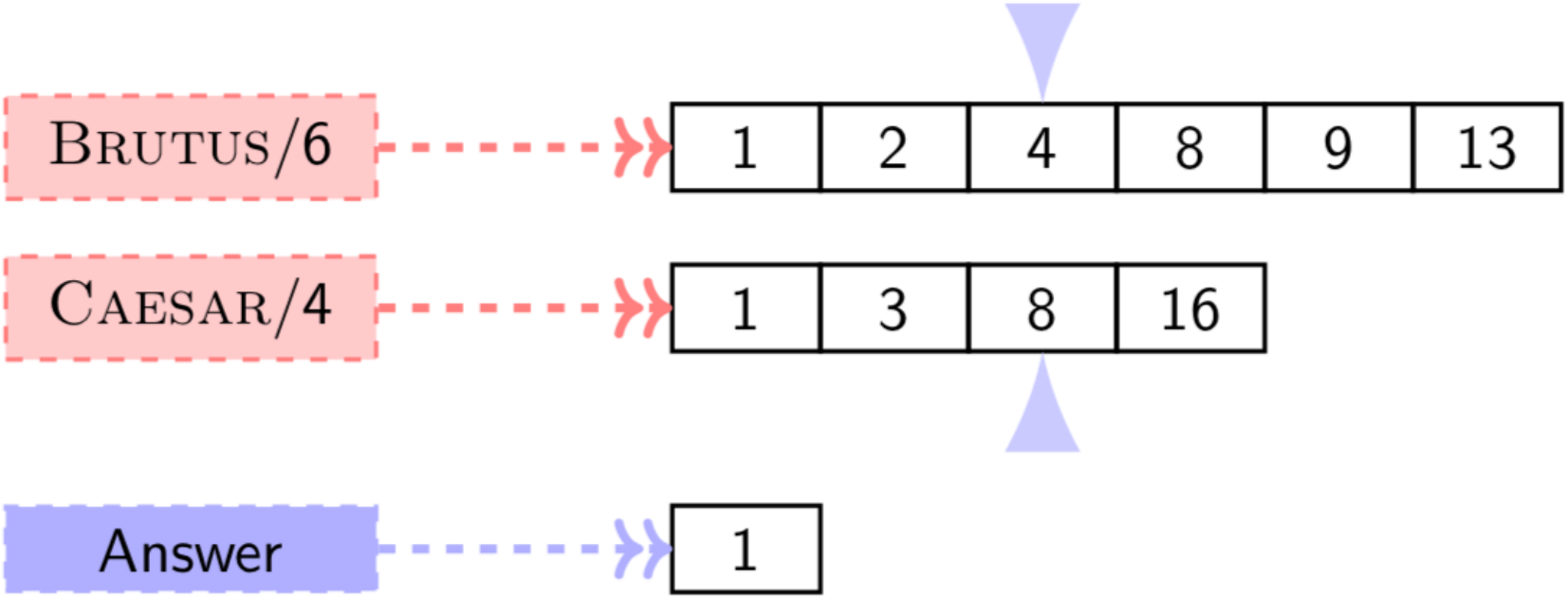
Boolean Retrieval

Boolean Retrieval with Inverted Index



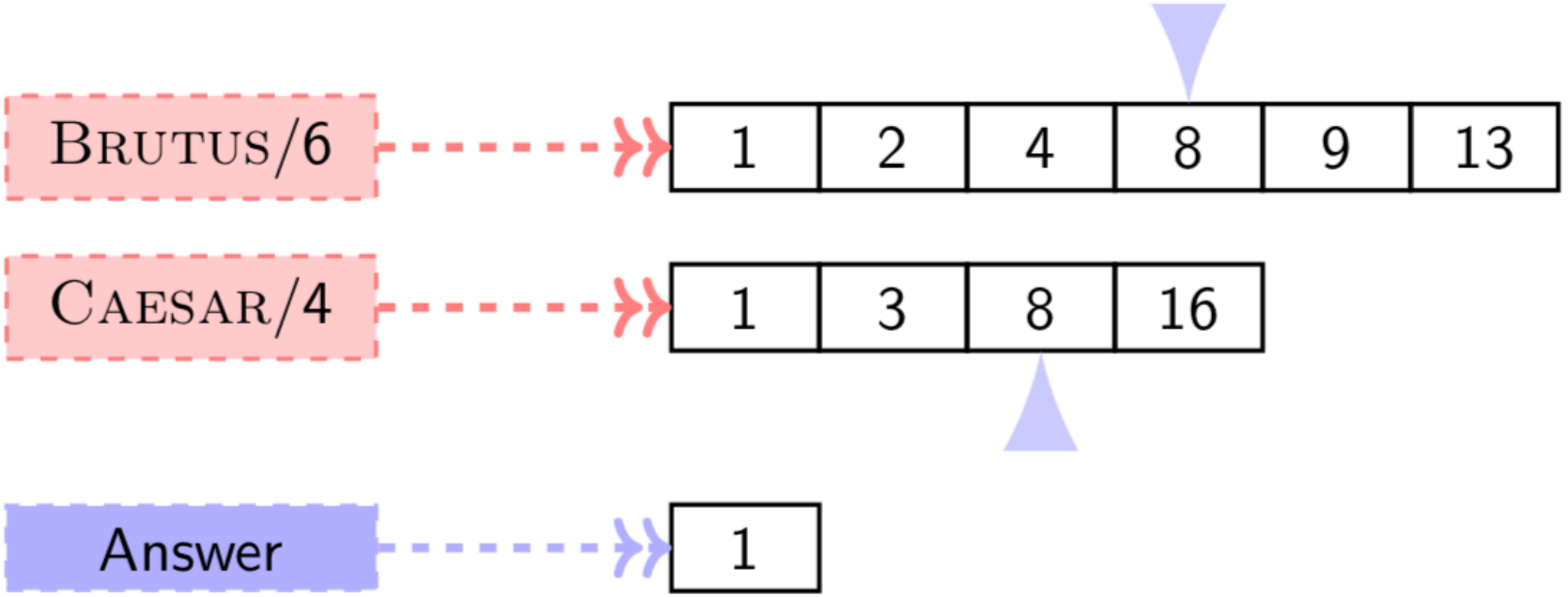
Boolean Retrieval

Boolean Retrieval with Inverted Index



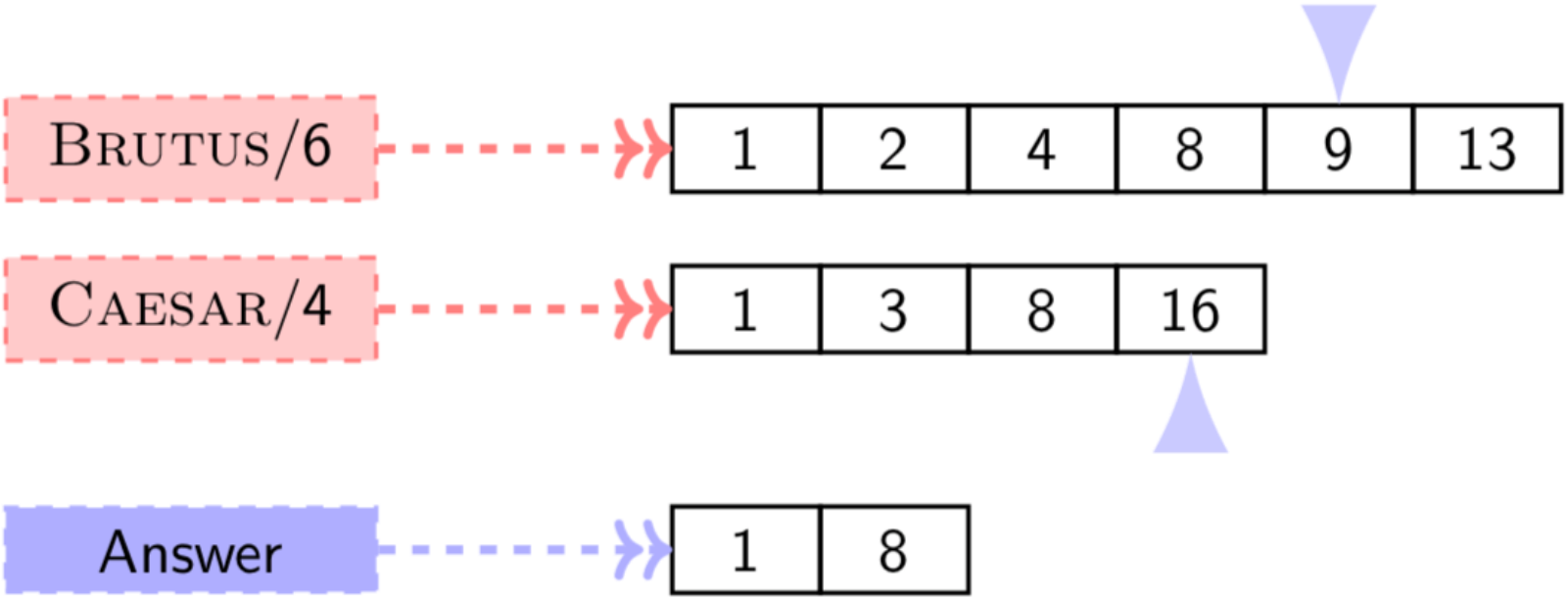
Boolean Retrieval

Boolean Retrieval with Inverted Index



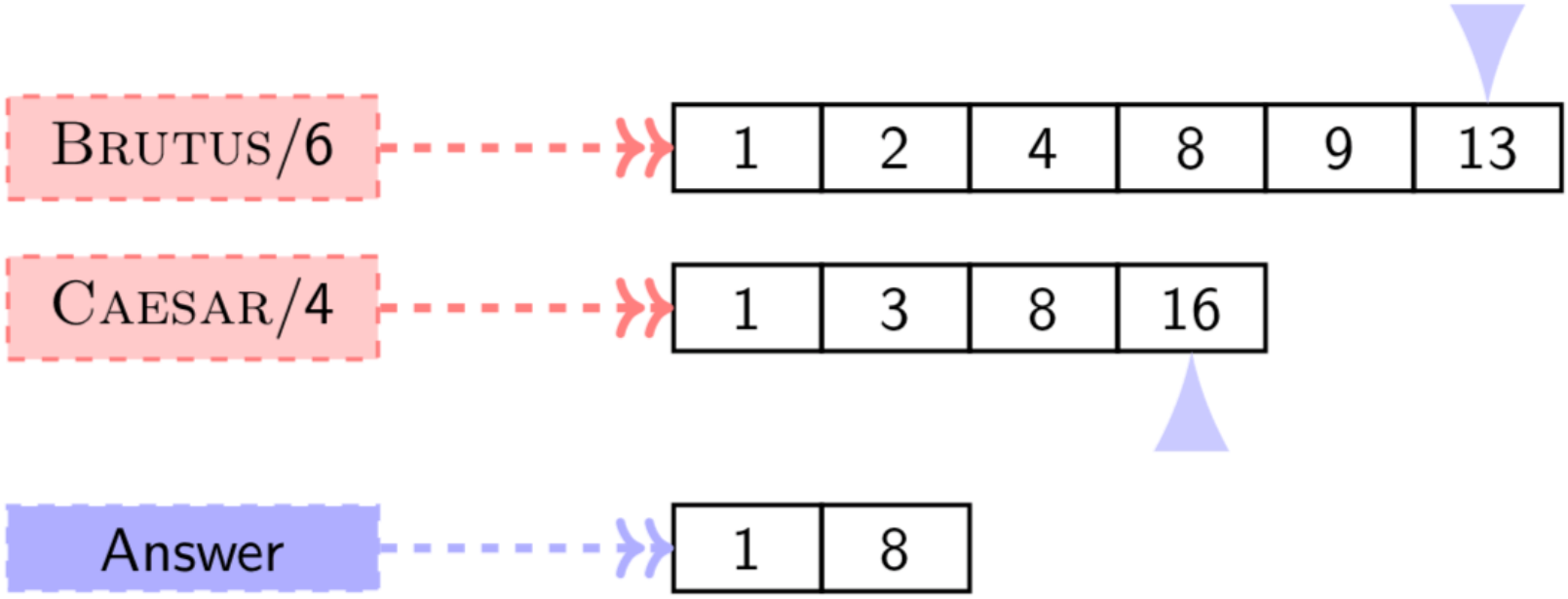
Boolean Retrieval

Boolean Retrieval with Inverted Index



Boolean Retrieval

Boolean Retrieval with Inverted Index



Boolean Retrieval

Boolean Retrieval with Inverted Index

- Can answer any query which is a Boolean expression: AND, OR, NOT
- Precise: each document matches, or not
- Extended Boolean allows more complex queries
- Primary commercial search for 30+ years, and is still used

Australian National University

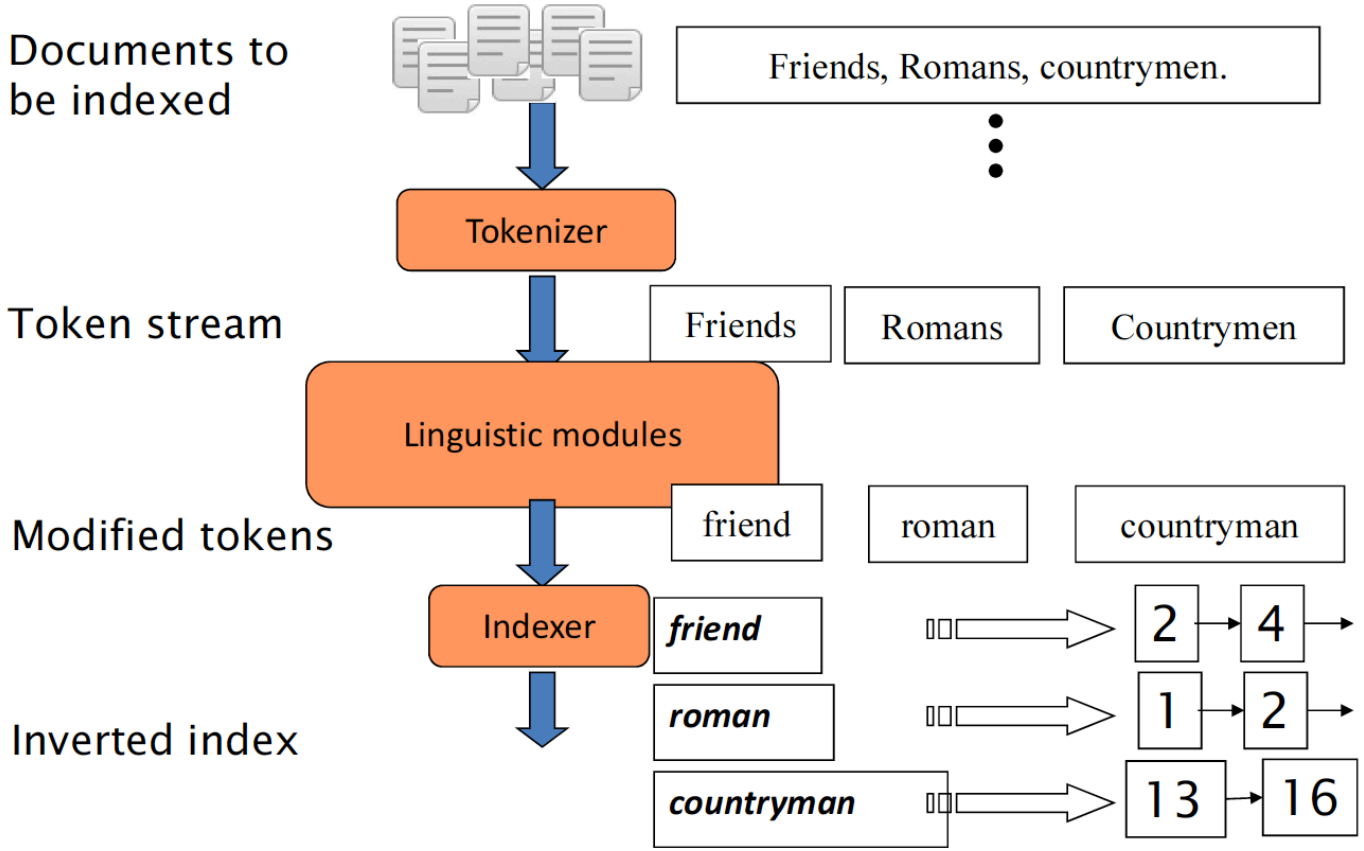
Outline

- Introduction to IR
  - What is information retrieval
  - Why information retrieval
  - How to perform information retrieval
  - IR vs NLP
- Boolean retrieval
  - Indexing, querying, and retrieval procedures
  - Term-document incidence matrix
  - Inverted index
  - Boolean retrieval with inverted index
- Initial stages of text processing

Text Preprocessing

Initial Stages of Text Processing

- So far we assumed that we can easily scan terms from a document. But the scanning consists of following steps:
  - Tokenisation, stopwords removal, and normalisation



Text Preprocessing

Tokenisation

- The task of dividing text into tokens
- Token: word, number, punctuation mark, and other symbols
- Simply chopping by whitespace and throwing punctuation are not enough
  - Locations, e.g. `New York, San Francisco`
  - Phone numbers, e.g. `(800) 234 2333`
  - Dates, e.g. `Mar 11, 1983`
  - `LOWERCASE` and `LOWER CASE`
- Periods:
  - `Ph.D.`, `google.com`
- Clitics:
  - `isn’t => is + n't` (not)
  - `n't` is a clitic (it can't occur independently, but functions like a word)
- Hyphenation:
  - `co-operate`, `Hewlett-Packard`
  - `most-visited => most visited`
  - `doc-ument` (line end)

Text Preprocessing

Tokenisation

- Tokenisation is a language dependent problem
- Alphabetic languages (e.g. English, French, Korean)
  - Words are separated by space
  - Deterministic methods are sufficiently accurate
    - Regular expressions for English: simple and efficient
    - e.g. `[a-zA-Z0-9]+-?\w+`
- Logographic writing systems (e.g. Mandarin, Japanese, Thai)
  - No spaces between words
  - Probabilistic methods are more accurate (e.g. CRF)
- Whatever method you use, always **do the same tokenisation** of document and query text

Text Preprocessing

Stopwords Removal

- Stop words usually refer to the most common words in a language, e.g. the, a, an, and, or, will, would, could
- Remove words in a predefined stop word list
- Or ignore the most frequent (e.g. top 100) words found in the training corpus
- In information retrieval
  - These words are not very useful in keyword search
  - Reduce the number of postings that a system has to store

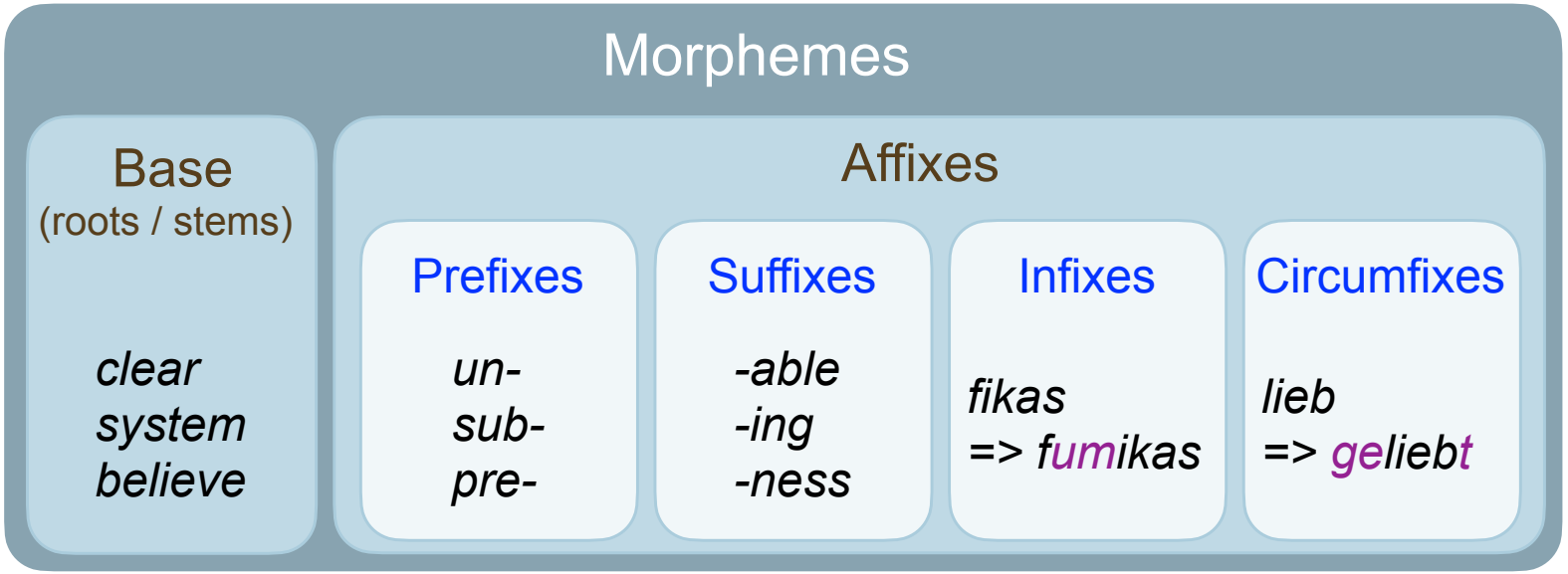
Normalisation

- The task of converting a words/tokens into standard format
- Normalisation based on **morphological analysis**: Stemming, Lemmatisation
- Keep equivalence class of terms: `U.S.A = USA = united states`
- Synonym list: `car = automobile`
- Capitalisation: `ferrari => Ferrari`
- Case-folding: `Automobile => automobile, ACT != act`

Text Preprocessing

Normalisation

- Morphology is the study of the way words are built up from smaller meaning-bearing units called morphemes
  - Morpheme: minimal meaning-bearing unit in a language
  - Root: morpheme that remains when all affixes are stripped from a complex word
  - root + affix => stem, stem + affixes => stem
  - Language dependent: some languages have very rich morphology (e.g. Czech, French), others poor (e.g. English)
- Stemming and lemmatisation are text normalisation approaches based on morphological analysis



Text Preprocessing

Stemming

- Stemming is the task of turning tokens into stems
- Stems are the same regardless of inflection, e.g. `{run, runs, running}` => `run`
- Stems need not be real words
- Stemming is usually a crude heuristic process that strips off suffixes, e.g. `studies` => stem: `studi`, suffix: `es`
- Algorithms (rule-based): e.g. lookup table, regular expressions, suffix-stripping

Table: Porter Stemmer Example

Rule	Example		
Replace `ies` with `i`	`ponies`	⇒	`poni`
Replace `y` with `i`	`pony`	⇒	`poni`
Remove `ing`	`falling`	⇒	`fall`
Remove `s`	`dogs`	⇒	`dog`
Remove `es`	`replaces`	⇒	`replac`
Remove `e`	`replace`	⇒	`replac`

Lemmatisation

- Lemmatisation is the task of turning words into lemmas, which are entries in a dictionary e.g. the word "`better`" has "`good`" as its lemma
- Requires knowledge of the context (e.g. the intended Part-of-Speech of the word)
- Results depending on the dictionary used
- Usually slower and computationally more expensive than stemming

Text Preprocessing

Tools

- NLTK - https://www.nltk.org
- spaCy - https://spacy.io
- OpenNLP - https://opennlp.apache.org
- Stanford CoreNLP - https://stanfordnlp.github.io/CoreNLP
- Stemmer
  - Porter Stemmer - https://tartarus.org/martin/PorterStemmer
  - Snowball Stemmers - https://snowballstem.org
  - Demo - http://text-processing.com/demo/stem

Summary

- Introduction to IR
  - What is information retrieval
  - Why information retrieval
  - How to perform information retrieval
  - IR vs NLP
- Boolean retrieval
  - Indexing, querying, and retrieval procedures
  - Term-document incidence matrix
  - Inverted index
  - Boolean retrieval with inverted index
- Initial stages of text processing

References

- Chapter 1, Introduction to Information Retrieval
- Chapter 2, Speech and Language Processing
- Some lecture slides are from:
  - Hongning Wang CS 4501/6501: Information retrieval, CS@UVa
  - Pandu Nayak and Prabhakar Raghavan, CS276, Information Retrieval and Web Search, Stanford University