Université Jean Monnet, Saint-Étienne
Faculté des Sciences et Techniques de Saint-Étienne
Laboratoire Hubert Curien, UMR CNRS 5516

Recherche d'Information (RI)

Mathias Géry
Mathias.Gery@univ-st-etienne.fr

Chapter 1
IR Model & Boolean Retrieval

1.1: Information Retrieval
1.2: IR Model & IR System
1.3: Indexing: inverted index
1.4: Querying: boolean 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:
  - E-mail search,
  - Searching your laptop,
  - Corporate knowledge bases,
  - Legal information retrieval,
  - Etc.

- IR also covers:
  - Routing (and Filtering)
  - Classification / Categorization
  - Clustering
  - Information Extraction
  - Recommendation
  - Question Answering

Data Retrieval vs. Information Retrieval

- Information Retrieval / 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.
- Data Retrieval / Structured data:
  - Tends to refer to information in “tables”.
  - Typically allows numerical range and exact match (for text) queries, e.g., Salary < 60000 AND Manager = ‘Smith’.

Employee	Manager	Salary
Smith	Jones	50000
Chang	Smith	60000
Ivi	Smith	50000

Data Retrieval (DR) vs. Information Retrieval (IR)

	DR	IR
Answer	record (data)	document reference
Model	deterministic	probabilistic
Query	accurate, complete, non ambiguous	fuzzy, incomplete, ambiguous
Query language	artificial	natural
Success criteria	exactitude, efficiency, ergonomy, integrability	satisfaction

Data types in IR

- Unstructured.
- Text data (news, reports, mails, etc.).

Nature	Size	Example
Text	1 Mb	A large novel
	500 Mb	An encyclopedia
	100 Gb	A library
	20 Tb	Library of Congress

- Non-text data (images, graphics, sounds, videos, etc.).

Nature	Size	Example
Sound	500 Mb	A symphony
Video	100 Gb	A movie (raw)
Image	1 Pb	Numerized Library of Congress

Data Size

- Three scales:
  1. Web: billions of documents stored on millions computers
  2. Enterprise
  3. Personal data

- Examples (source 2016):
  - Google: ~100 PB a day; 1+ million servers (est. 15-20 Exabytes stored).
  - Wayback Machine: 15+ PB + 100+ TB/month.
  - Facebook: 300+ PB of user data + 600 TB/day.
  - YouTube: ~1000 PB video storage + 4 billions views/day.
  - CERN's Large Hadron Collider: 15 PB/year.
  - NSA: ~2+ Exabytes stored.
- 2024: ~463 exabytes / day.
  - (1 exabyte = 1 000 000 Tb).

640K ought to be enough for anybody.
Bill.

Difficulties in IR

- Data Size.
- Unstructured data: semantic difficult to catch.
- All and every domain.
- User diversity.
- Difficult to know the actual information need.
- Distribution and multiplicity of information sources.
- Both **efficiency** and **effectiveness** are concerned.

Chapter 1
IR Model & Boolean Retrieval

Université Jean Monnet Saint-Étienne

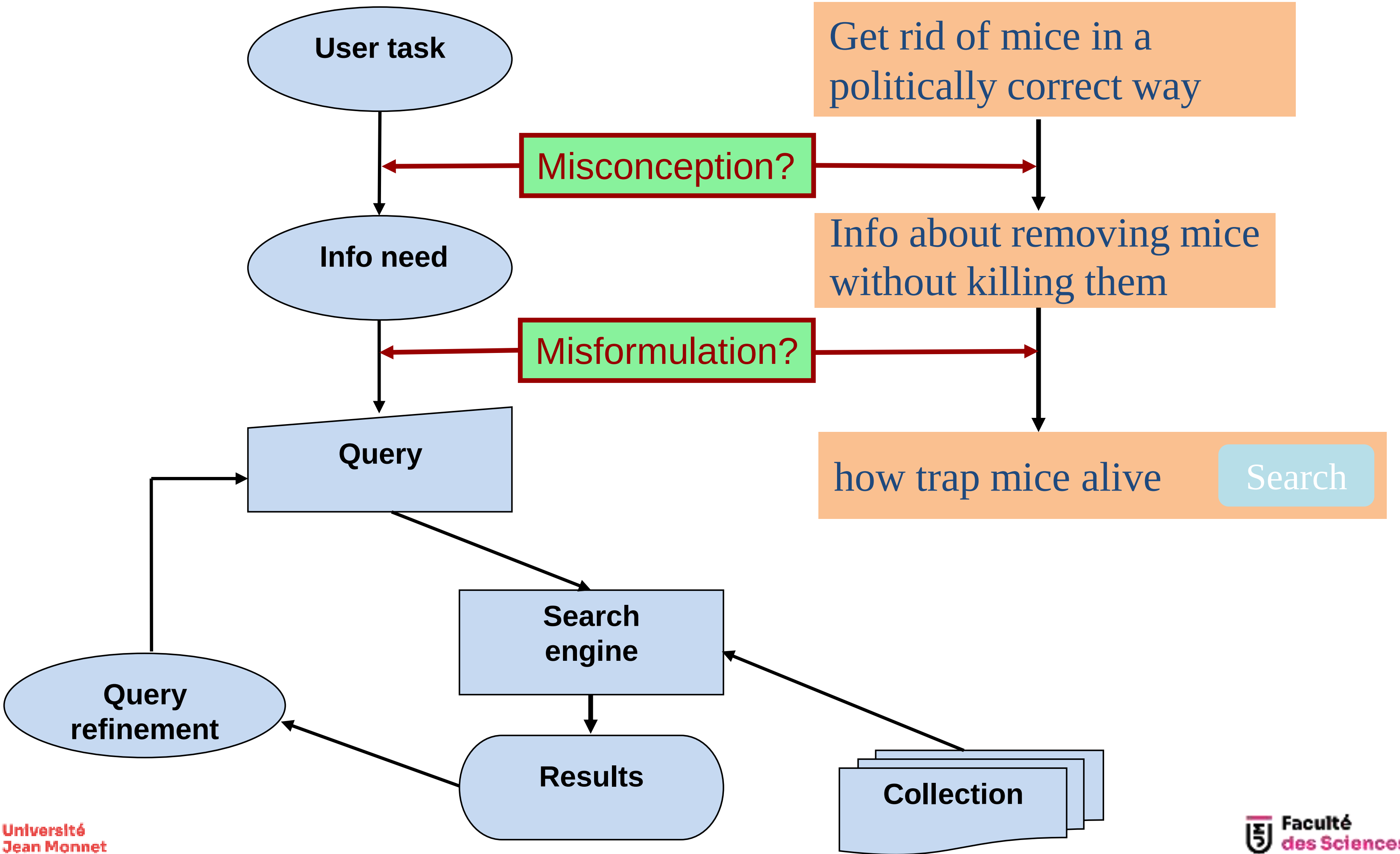
Faculté des Sciences et Techniques Saint-Étienne

Basic assumptions of IR

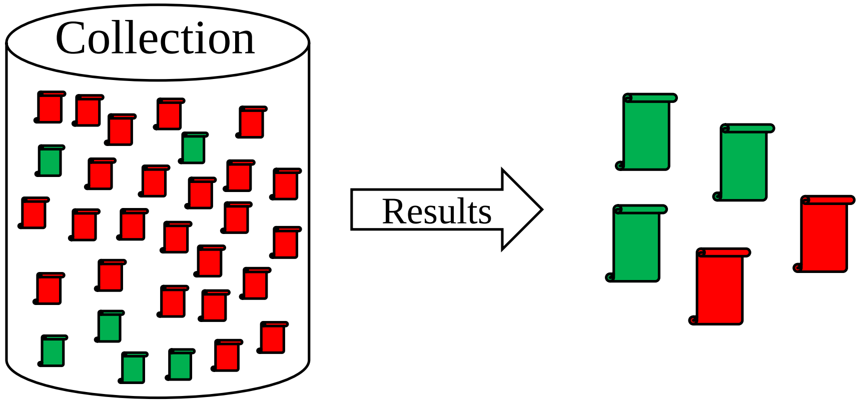
- **Collection**: A set of documents:
  - Assume it is a static collection.

- **Goal**: Retrieve documents with information that is **relevant** to the user's **information need** (… and helps the user complete a **task**).

Classic search model



How good are the retrieved docs?



Binary relevance: **yes** / **no**

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

  60%

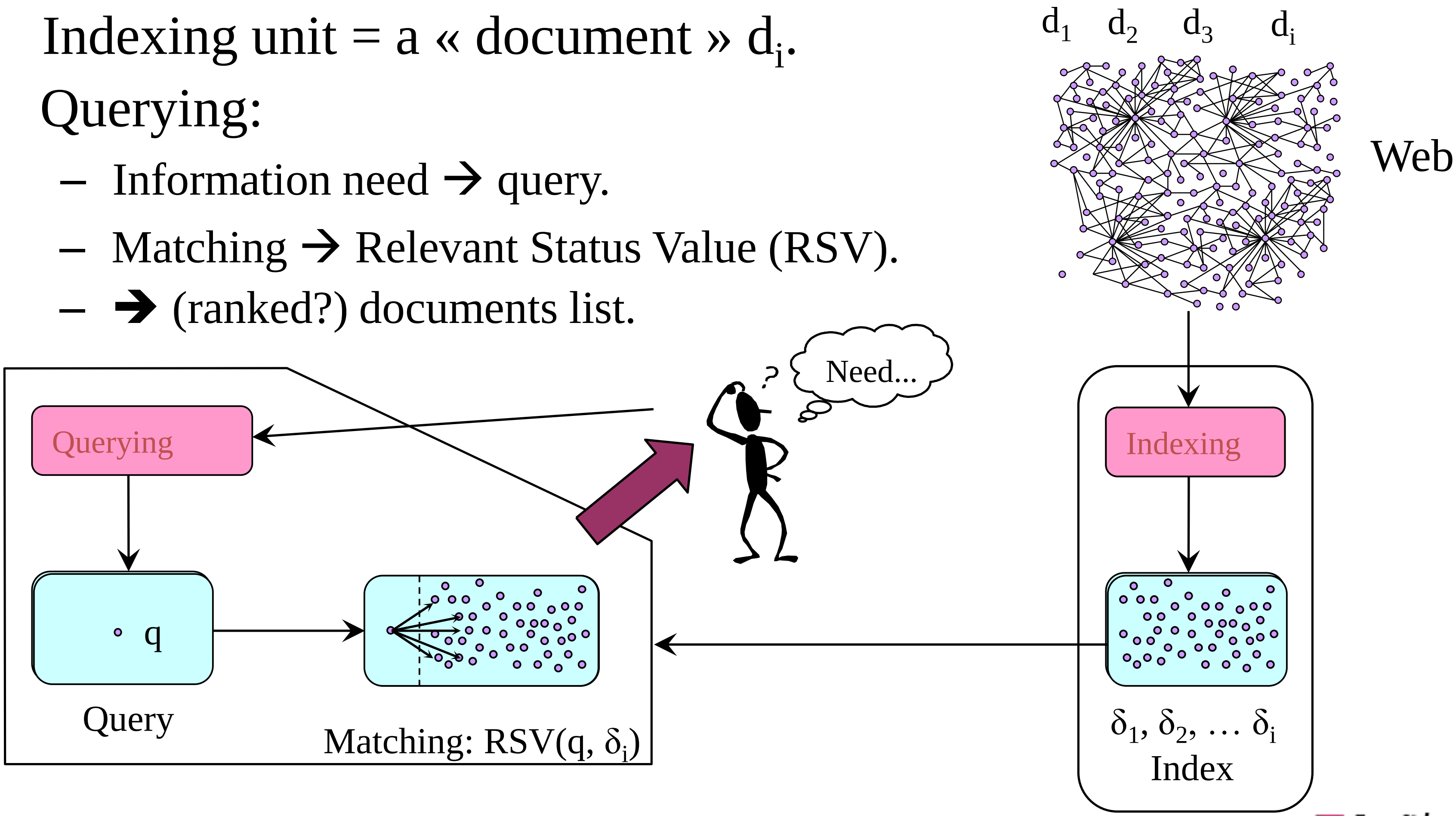
- ***Recall***: Fraction of relevant docs in collection that are retrieved.

  50%

- More precise definitions and measurements to follow later.

Classic IR model

- Collection of documents d_1, d_2, d_3, …
- Indexing unit = a « document » d_i.
- Querying:
  - Information need → query.
  - Matching → Relevant Status Value (RSV).
  - ➔ (ranked?) documents list.



IR System (IRS)

- Definition and creation of the collection.
- Matching function choice.
- Query language choice and definition.
- Users choice and definition:
  - Knowledge of IRS.
  - Kind of information needs.
  - Expertise.
- Document indexation.

Chapter 1
IR Model & Boolean Retrieval

1.1: Information Retrieval
1.2: IR Model & IR System
1.3: Indexing: inverted index
1.4: Querying: boolean retrieval

Unstructured data in 1620

Wikipedia

- Query: which plays of Shakespeare,
  - contain the words **Brutus** *AND* **Caesar**
  - but *NOT* **Calpurnia**?

- Why not:

  grep brutus Shakespeare-plays.txt | grep caesar | grep –v calpurnia

- Efficiency?
  - Slow (for large corpora).
- Flexibility?
  - Other operations (e.g., find the word ***Romans*** near ***countrymen***) not feasible.
- Ranking?
  - No ranked retrieval (best documents to return).

Université Jean Monnet Saint-Étienne

Faculté des Sciences et Techniques Saint-Étienne

Term-document incidence matrices

Query: ***Brutus*** *AND* ***Caesar*** *BUT NOT* ***Calpurnia***

	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

- Document: set of keywords.
- Document: vector of *{0,1}*.

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**

	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

Bigger collections

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

- Occupation ratio = 0,2%
  - 1 000 000 x 500 000 = 500 000 000 000 values (0's and 1's)
  - But "only" 1 000 000 x 1 000 = 1 000 000 000 x 1's

- What's a better representation?
  - We only record the 1 positions: **inverted index!**

Inverted index

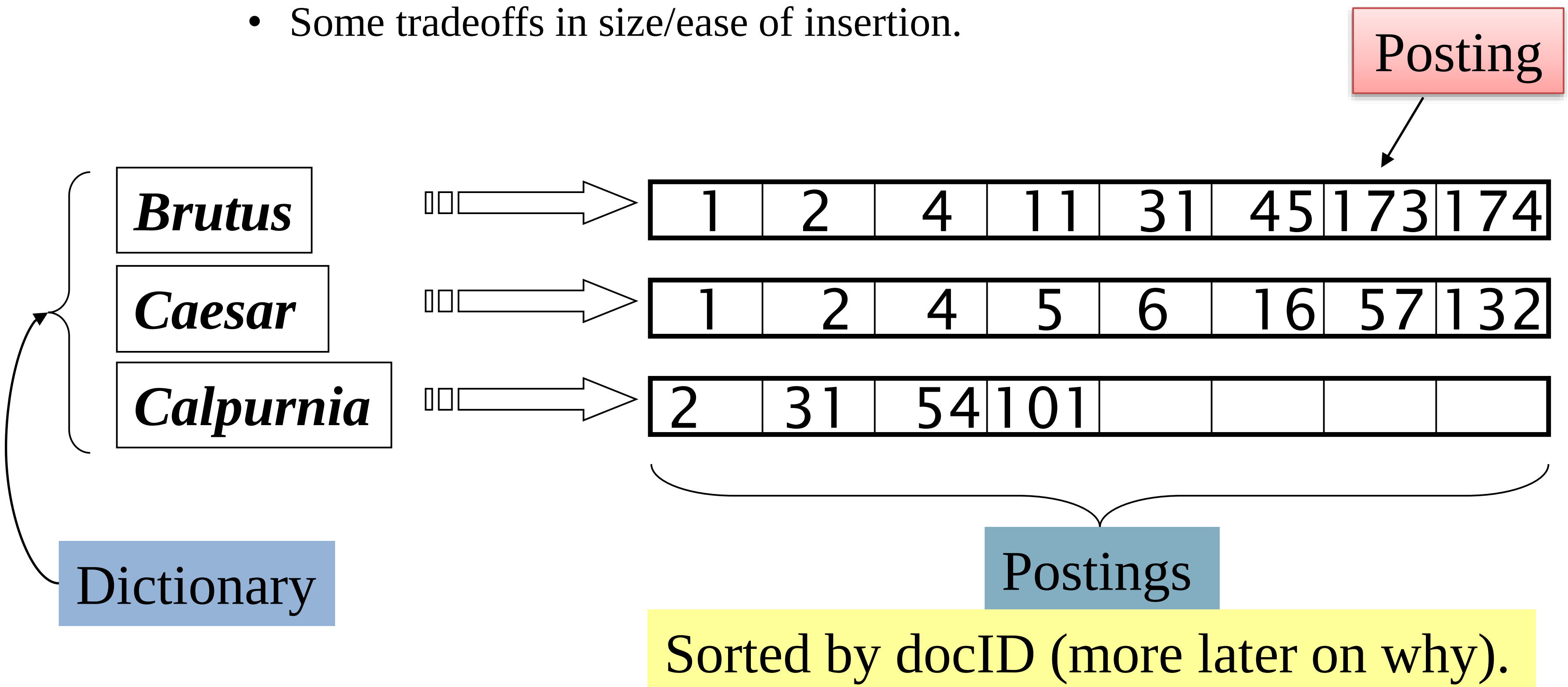
- The key data structure underlying modern IR.
- For each term *t*, we must store a list of all documents that contain *t*.
  - Identify each doc by a **docID** (a document serial number).
- Can we used fixed-size arrays for this?

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

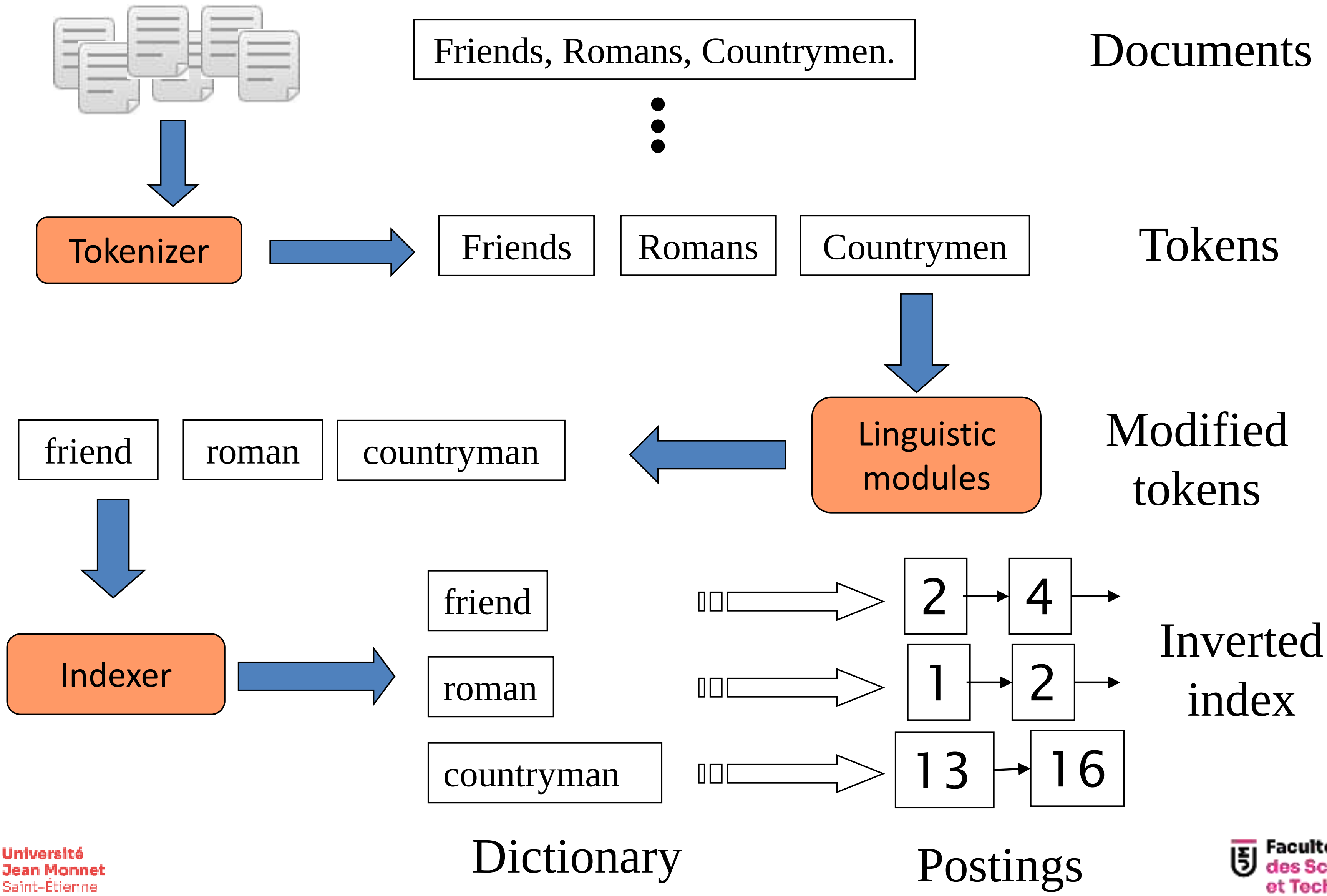
What happens if the word ***Caesar*** is added to document 14?

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.



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:
  - We may wish different forms of a root to match
    - ***authorize***, ***authorization***
- Stop words
  - We may omit very common words:
    - ***the, a, to, of***

Indexer steps: Token sequence

- Input: Set of documents.
- Output: Sequence of (modified token, DocID) pairs.

Doc 1	Doc 2
I did enact Julius Caesar I was killed i' the Capitol; Brutus killed me.	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

Université Jean Monnet Saint-Étienne

Indexer steps: Sort

- Sort by terms.
- And then docID.

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.

Why frequency?
Will discuss later.

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

Lists of docIDs

Terms and counts

Pointers

IR system implementation:

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

Chapter 1
IR Model & Boolean Retrieval

1.1: Information Retrieval
1.2: IR Model & IR System
1.3: Indexing: inverted index
1.4: Querying: boolean retrieval

Query processing with an inverted index

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

Query processing: AND

- Consider processing the query:

  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***

Université Jean Monnet Saint-Étienne

Faculté des Sciences et Techniques Saint-Étienne

The merge

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

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

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

List lengths = x and y ➔ merge = $O(x+y)$ operations.

<u>Crucial</u>: postings sorted by docID.

Université Jean Monnet Saint-Étienne

Faculté des Sciences et Techniques Saint-Étienne

Intersecting two postings lists

- A "merge" algorithm:

```
INTERSECT(p1, p2)
answer ← ⟨ ⟩
while p1 ≠ NIL and p2 ≠ NIL
do if docID(p1) = docID(p2)
      then ADD(answer, docID(p1))
           p1 ← next(p1)
           p2 ← next(p2)
      else if docID(p1) < docID(p2)
              then p1 ← next(p1)
              else p2 ← next(p2)
return answer
```

Boolean queries: Exact match

- The Boolean retrieval model is being able to ask a query that is a Boolean expression:
  - Boolean 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 commercial IR Systems 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 (multiple data sources) added 2010.

- Tens of terabytes of data; ~700,000 users; millions of searches / day.

- Majority of users *still* use (and need!) boolean queries.

Example: WestLaw

- Long, precise queries; proximity operators; incrementally developed; not like web search.
- Example queries:
  - What is the statute of limitations in cases involving the federal tort claims act?
    - LIMIT! /3 STATUTE ACTION /S FEDERAL /2 TORT /3 CLAIM
  - Requirements for disabled people to be able to access a workplace
    - disabl! /p access! /s work-site work-place (employment /3 place)
  - /3 = within 3 words, /s = in same sentence, /p = in same paragraph
  - SPACE is disjunction, not conjunction!
- Many professional searchers still like Boolean search
  - You know exactly what you are getting.
  - But that doesn't mean it actually works better….

Boolean queries: 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***

Boolean queries: optimization

- Process in order of increasing frequency:
  - 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						

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

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

Université Jean Monnet Saint-Étienne

Faculté des Sciences et Techniques Saint-Étienne

More general optimization

- Query:

  *(**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.
- Process in increasing order of *OR* sizes.