Introduction to

Information Retrieval

01 Boolean retrieval

BK인턴 유승미

Contents

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).
  - Now, people engage in information retrieval every day when they use a web search engine or search their email.

Unstructured & Semistructured

- "unstructured data" ⟺ structured data

  refers to data which does not have clear, semantically overt, easy-for-a-computer structure.

  - In reality, almost no data are truly "unstructured"

- IR is also used to facilitate "semistructured" search.

  (such as finding a document where the title contains Java and the body contains threading.)

The three prominent scales

1. Web Search: Involves searching billions of documents across millions of computers.
   - Key issues: document gathering, efficient system building, exploiting hypertext, and dealing with manipulation by site providers to boost rankings.
2. Personal Information Retrieval: Integrated into consumer operating systems.
   like Mac OS X Spotlight or Windows Vista's Instant Search
   - Challenges: handling various document types and ensuring the system is lightweight and maintenance-free for single-machine use.
3. Enterprise, institutional, and domain-specific search:
   Applied to collections like corporate documents, patent databases, or research articles, typically stored on centralized systems with dedicated machines for search.
   - coverage of parallel and distributed search in web-scale systems is limited

An example information retrieval problem

ex) a play of Shakespeare's → the Boolean retrieval model

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

- a binary 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	
…							

110100 AND 110111 AND 101111 = 100100

Answer

Antony and Cleopatra & Hamlet

Source: https://nlp.stanford.edu/IR-book/information-retrieval-book.html

Information Retrieval

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

How good are the retrieved docs?

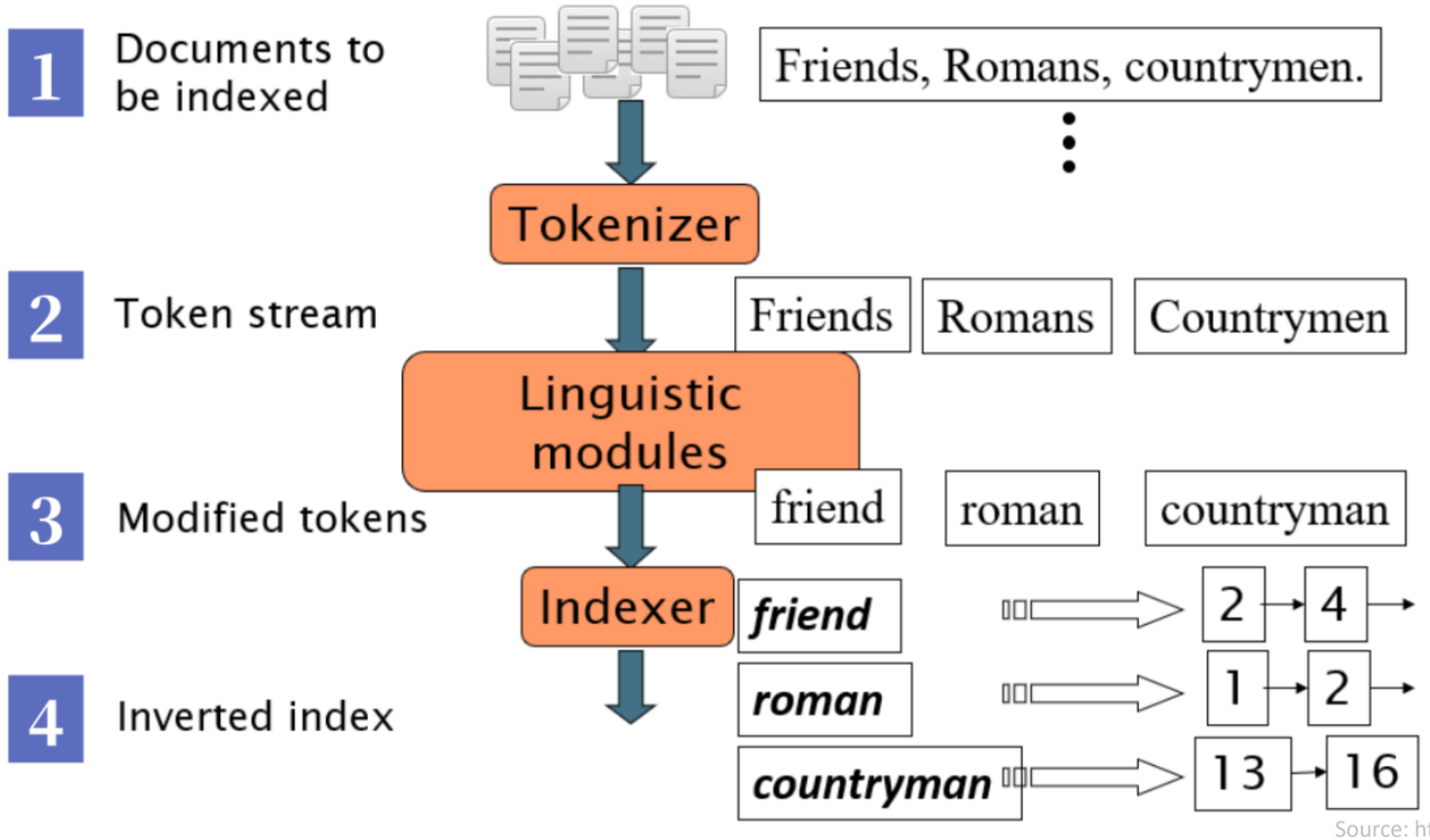
Precision Fraction of retrieved docs that are relevant to the user's information need

Recall Fraction of relevant docs in collection that are retrieved

A first take at building an inverted index

To gain the speed benefits of indexing at retrieval time, we have to build the index in advance.

Inverted Index



Source: https://web.stanford.edu/class/cs276/

Indexer steps: Token sequence

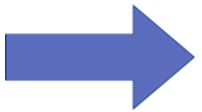
- Sequence of (Modified token, Document ID) pairs.

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

Source: https://web.stanford.edu/class/cs276/

Indexer steps: Sort

- Sort by terms
  - alphabetically, 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

Source: https://web.stanford.edu/class/cs276/

Indexer steps: Dictionary & Postings

- Multiple term entries in a single document are merged.
- Split into Dictionary and Postings
  * Doc. frequency information is added
  * The postings are secondarily sorted by docID

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

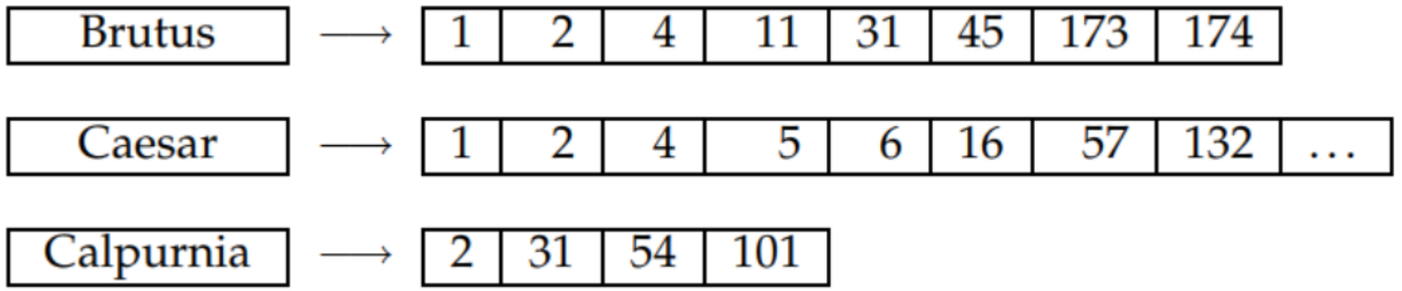
Source: https://web.stanford.edu/class/cs276/

Inverted index

- In the resulting index,
  we pay for storage of both the dictionary and the postings lists.

ex) 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



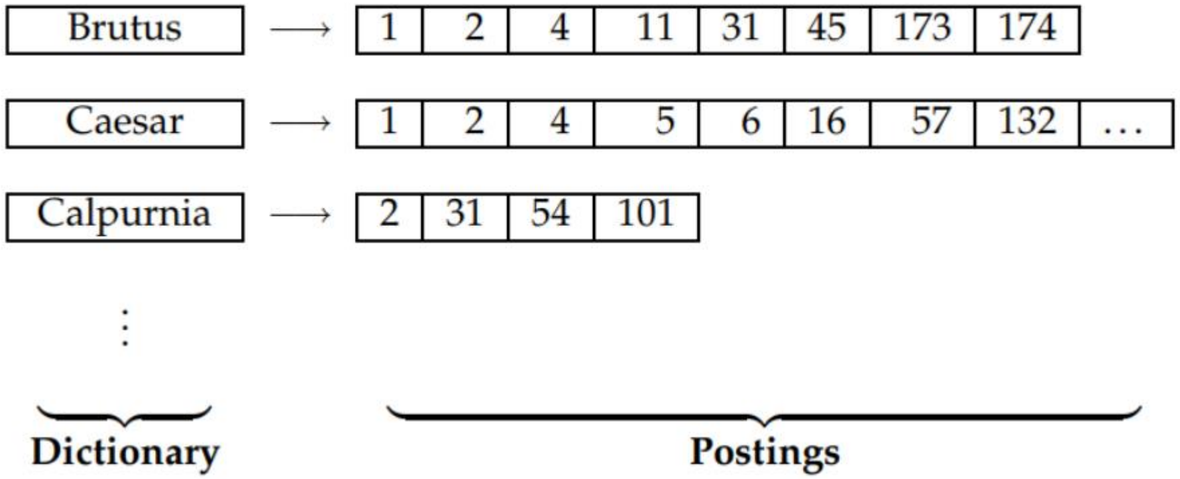
→ We can't used fixed-size arrays for this.

A first take at building an inverted index

- For an in-memory postings list, two good alternatives are singly linked lists [1] or variable length arrays. [2]

[1] facilitate easy insertion of documents and extend to advanced strategies like skip lists.

[2] more space-efficient and faster due to contiguous memory usage.



Source: https://nlp.stanford.edu/IR-book/information-retrieval-book.html

Query processing: AND

- Consider processing the simple conjunctive query:

  Brutus AND Calpurnia

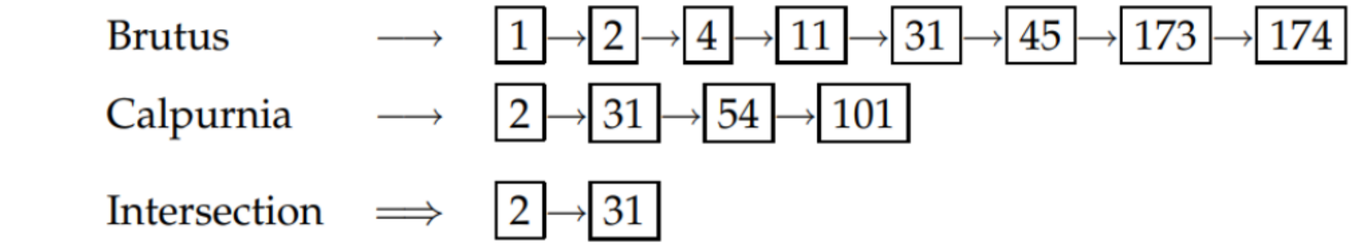
  1. Locate *Brutus* in the Dictionary
  2. Retrieve its postings.
  3. Locate Calpurnia in the Dictionary
  4. Retrieve its postings.
  5. “Merge” the two postings (intersect the document sets)

Query processing: AND

- Consider processing the simple conjunctive query:

  Brutus AND Calpurnia

```
INTERSECT(p1, p2)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then ADD(answer, docID(p1))
 5             p1 ← next(p1)
 6             p2 ← next(p2)
 7        else if docID(p1) < docID(p2)
 8                then p1 ← next(p1)
 9                else p2 ← next(p2)
10  return answer
```



► **Figure 1.5** Intersecting the postings lists for Brutus and Calpurnia from Figure 1.3.

► **Figure 1.6** Algorithm for the intersection of two postings lists p_1 and p_2.

Source: https://nlp.stanford.edu/IR-book/information-retrieval-book.html

Boolean Retrieval Model

- 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
- Contrasts with ranked retrieval models like the vector space model.
  - Users typically use free text queries in ranked models, typing one or more words.
- Boolean retrieval dominated until the early 1990s, especially among commercial providers.
  - Extended Boolean models included additional operators.

    * Term proximity operators require query terms to appear close to each other in documents.

Medinfo Lab
Seoul National University
Medical Informatics Laboratory

감사합니다.

Source: https://web.stanford.edu/class/cs276/
Source: https://nlp.stanford.edu/IR-book/information-retrieval-book.html