# Information Retrieval

Weike Pan

The slides are **adapted from those provided by Prof. Hinrich Schütze** at University of Munich (http://www.cis.lmu.de/~hs/teach/14s/ir/).

# Chapter 1 Boolean retrieval

- 1.1 An example information retrieval problem
- 1.2 A first take at building an inverted index
- 1.3 Processing Boolean queries
- 1.4 The extended Boolean model versus ranked retrieval
- 1.5 References and further reading

# Outline

- 1.1 An example information retrieval problem
- 1.2 A first take at building an inverted index
- 1.3 Processing Boolean queries
- 1.4 The extended Boolean model versus ranked retrieval
- 1.5 References and further reading

# 1.1 An example information retrieval problem

**Definition of 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).

# 1.1 An example information retrieval problem

**Boolean retrieval**

- The Boolean model is arguably the simplest model to base an information retrieval system on.
- Queries are Boolean expressions, e.g., CAESAR AND BRUTUS
- The search engine returns all documents that satisfy the Boolean expression.

# 1.1 An example information retrieval problem

**Does Google use the Boolean model?**

- On Google, the default interpretation of a query [w1 w2 ... wn] is w1 AND w2 AND ... AND wn

- Cases where you get hits that do not contain one of the wi:
  - anchor text
  - page contains a variant of wi (e.g., morphology, spelling correction, synonym)
  - long queries (i.e., n is large)
  - Boolean expression generates very few hits

# 1.1 An example information retrieval problem

- Simple Boolean vs. Ranking of result set
  - Simple Boolean retrieval returns matching documents in no particular order
  - Google (and most well designed Boolean engines) rank the result set -- they rank good hits (according to some estimator of relevance) higher than bad hits

# 1.1 An example information retrieval problem

**Term-document incidence matrix**

- Entry is 1 if term occurs. Entry is 0 if term doesn't occur.

| | Anthony and Cleopatra | Julius Caesar | The Tempest | Hamlet | Othello | Macbeth | ... |
|---|---|---|---|---|---|---|---|
| ANTHONY | 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 | |
| ... | | | | | | | |

# 1.1 An example information retrieval problem

**Incidence vectors**

- So we have a 0/1 vector for each term

- To answer the query BRUTUS AND CAESAR AND NOT CALPURNIA:

  - Step 1. Take the vectors for BRUTUS, CAESAR, and CALPURNIA

  - Step 2. Complement the vector of CALPURNIA

  - Step 3. Do a (bitwise) AND on the three vectors 110100 AND 110111 AND 101111 = 100100

# 1.1 An example information retrieval problem

**Bigger collections**

- Consider N=10^6 documents, each with about 1000 tokens -> total of 10^9 tokens
- On average 6 bytes per token, including spaces and punctuation -> size of document collection is about 6*10^9=6GB
- Assume there are M=500,000 distinct terms in the collection
- Notice that we are making a term/token distinction

# 1.1 An example information retrieval problem

**Can't build the incidence matrix**

- 500,000*10^6 = half a trillion 0s and 1s

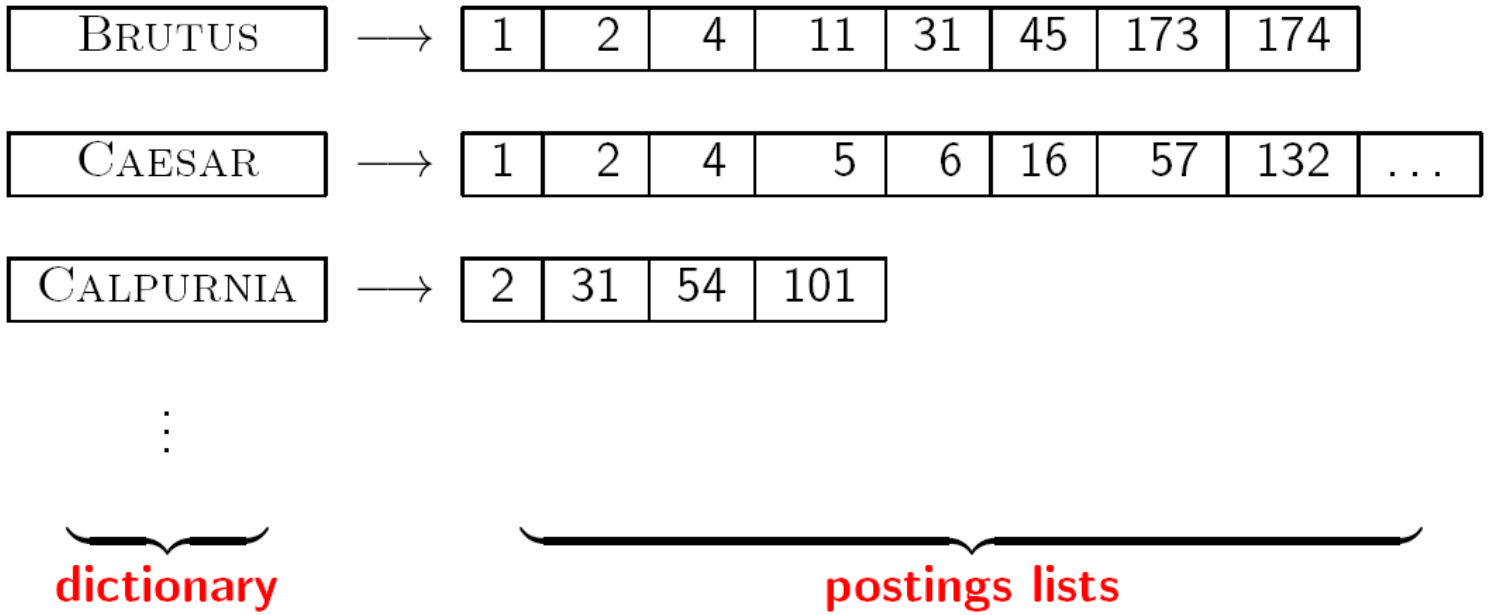
- But the matrix has no more than one billion 1s -> The incidence matrix is extremely sparse

- What is a better representation?
  - We only record the 1s

# 1.1 An example information retrieval problem

**Inverted index**

- For each term t, we store a list of documents that contain t.



# Outline

- 1.1 An example information retrieval problem
- 1.2 A first take at building an inverted index
- 1.3 Processing Boolean queries
- 1.4 The extended Boolean model versus ranked retrieval
- 1.5 References and further reading

# 1.2 A first take at building an inverted index

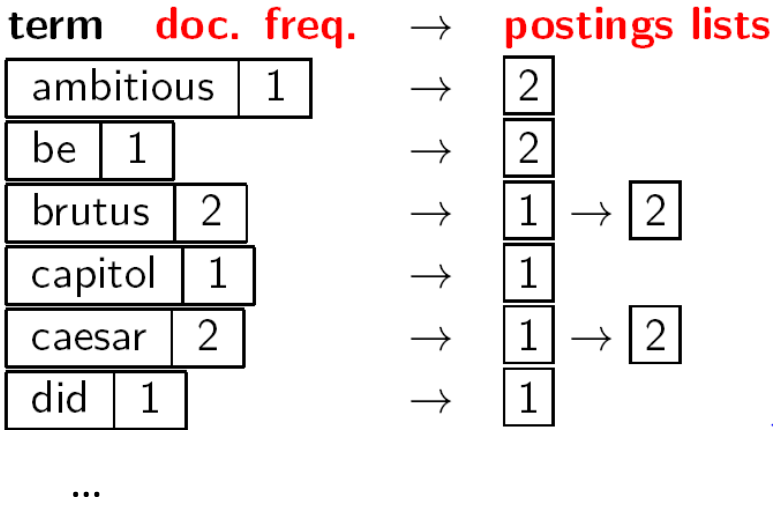
**Inverted index construction (1/2)**

- Step 1. Collect the documents to be indexed
- Step 2. Tokenize the text, turning each document into a list of tokens
- Step 3. Do linguistic preprocessing, producing a list of normalized tokens, which are the indexing terms
- Step 4. Index the documents that each term occurs in by creating an inverted index, consisting of a dictionary and postings lists

# 1.2 A first take at building an inverted index

**Inverted index construction (2/2)**

- 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.



# 1.2 A first take at building an inverted index

**Other issues**

- Index construction: how can we create inverted indexes for large collections?
- How much space do we need for dictionary and index?
- Index compression: how can we efficiently store and process indexes for large collections?
- Ranked retrieval: what does the inverted index look like when we want the "best" answer?

# 1.2 A first take at building an inverted index

**Unstructured data in 1650**

- 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.

# 1.2 A first take at building an inverted index

- Why is grep not the solution?
  - Slow (for large collections)
  - grep is line-oriented, IR is document-oriented
  - "NOT CALPURNIA" is non-trivial
  - Other operations (e.g., find the word ROMANS near COUNTRYMAN) are not feasible

# Outline

- 1.1 An example information retrieval problem
- 1.2 A first take at building an inverted index
- 1.3 Processing Boolean queries
- 1.4 The extended Boolean model versus ranked retrieval
- 1.5 References and further reading

# 1.3 Processing Boolean queries

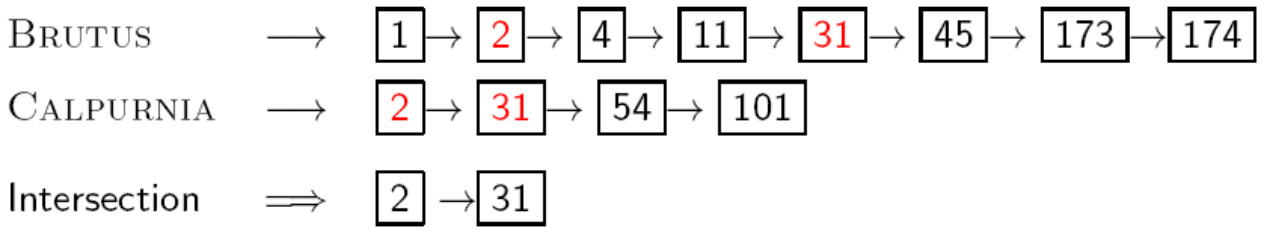
**Simple conjunctive query (two terms)**

- Consider the query: BRUTUS AND CALPURNIA

- To find all matching documents using inverted index:
  - Locate BRUTUS in the dictionary
  - Retrieve its postings list from the postings file
  - Locate CALPURNIA in the dictionary
  - Retrieve its postings list from the postings file
  - Intersect the two postings lists
  - Return the intersection to the user

# 1.3 Processing Boolean queries

**Intersecting two postings lists (1/2)**




- This is linear in the length of the postings lists.
- Note: This only works if postings lists are sorted.

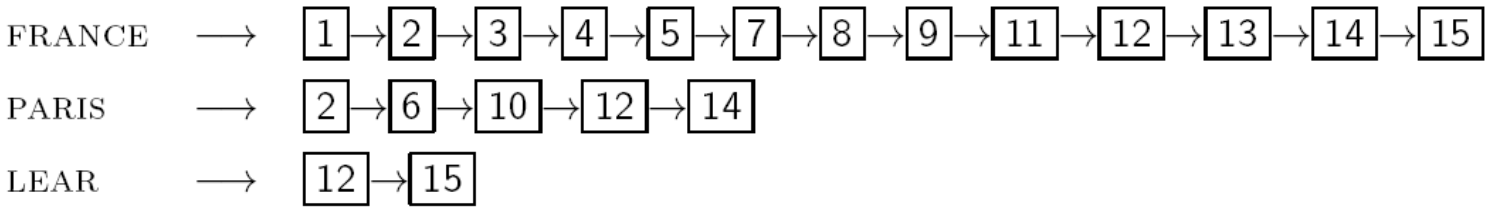
# 1.3 Processing Boolean queries

**Intersecting two postings lists (2/2)**

```
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
```

# 1.3 Processing Boolean queries

**Exercise**




- Compute the hit list for ((PARIS AND NOT FRANCE) OR LEAR)

# 1.3 Processing Boolean queries

**Boolean retrieval model: Assessment**

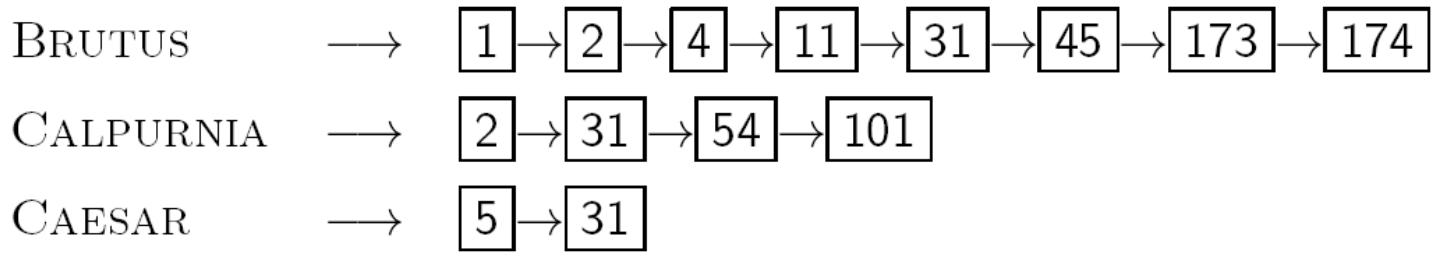
- The Boolean retrieval model can answer any query that is a Boolean expression
  - Boolean queries are queries that use AND, OR and NOT to join query terms
  - Views each document as a set of terms
  - Is precise: Document matches condition or not
- Primary commercial retrieval tool for 3 decades
- Many professional searchers (e.g., lawyers) still like Boolean queries, because you know exactly what you are getting
- Many search systems you use are also Boolean: email, intranet, etc

# 1.3 Processing Boolean queries

**Query optimization**

- Consider a query that is an AND of n terms, n>2
- For each of the terms, get its postings list, then AND them together

- Example query: BRUTUS AND CALPURNIA AND CAESAR
- What is the best order for processing this query?
- Simple and effective optimization: Process in order of increasing frequency



# 1.3 Processing Boolean queries

**More general optimization**

- Example query: (MADDING OR CROWD) AND (IGNOBLE OR STRIFE)
- Get frequencies for all terms
- Estimate the size of each OR by the sum of its frequencies (conservative)
- Process in increasing order of OR sizes

# Summary

- 1.1 An example information retrieval problem
- 1.2 A first take at building an inverted index
- 1.3 Processing Boolean queries
- 1.4 The extended Boolean model versus ranked retrieval
- 1.5 References and further reading