Boolean Retrieval[1]

September, 2009

[1] Vorlage: Folien von M. Schütze zu [1]

Boolean Retrieval

- Fragen sind Boolesche Ausdrücke, z.B.: Caesar AND Brutus
- Die Suchmaschine gibt **alle** Dokumente zurück, die der Anfrage entsprechen.

Does Google use the Boolean model?

Outline

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.
- Why is grep not the solution?

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.
- Why is grep not the solution?
  - Slow (for large collections)
  - " NOT Calpurnia" is non-trivial
  - Other operations (e.g., find the word Romans near countryman) not feasible
  - Ranked retrieval (best documents to return) – focus of later lectures, but not this one

Term-document incidence matrix

	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	

...

Entry is 1 if term occurs. Example: Calpurnia occurs in Julius Caesar.
Entry is 0 if term doesn't occur. Example: Calpurnia doesn't occur in The tempest.

Term-document incidence matrix

	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	

...

Entry is 1 if term occurs. Example: Calpurnia occurs in Julius Caesar.

Entry is 0 if term doesn't occur. Example: Calpurnia doesn't occur in The tempest.

Term-document incidence matrix

	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	

...

Entry is 1 if term occurs. Example: Calpurnia occurs in Julius Caesar.

Entry is 0 if term doesn't occur. Example: Calpurnia doesn't occur in The tempest.

Incidence vectors

- So we have a 0/1 vector for each term.
- To answer the query Brutus AND Caesar AND NOT Calpurnia:

Incidence vectors

- So we have a 0/1 vector for each term.
- To answer the query Brutus AND Caesar AND NOT Calpurnia:
  - Take the vectors for Brutus, Caesar, and Calpurnia
  - Complement the vector of Calpurnia
  - Do a (bitwise) AND on the three vectors
  - 110100 AND 110111 AND 101111 = 100100

0/1 vector for Brutus

	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	
...							

0/1 vector for Brutus

	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	
...							

0/1 vector for Brutus

	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	
...							

Answers to query

Anthony 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 $N = 10^6$ documents, each with about 1000 tokens

Bigger collections

- Consider $N = 10^6$ documents, each with about 1000 tokens
- On average 6 bytes per token, including spaces and punctuation $\Rightarrow$ size of document collection is about 6 GB

Bigger collections

- Consider $N = 10^6$ documents, each with about 1000 tokens
- On average 6 bytes per token, including spaces and punctuation $\Rightarrow$ size of document collection is about 6 GB
- Assume there are $M = 500{,}000$ distinct terms in the collection

Bigger collections

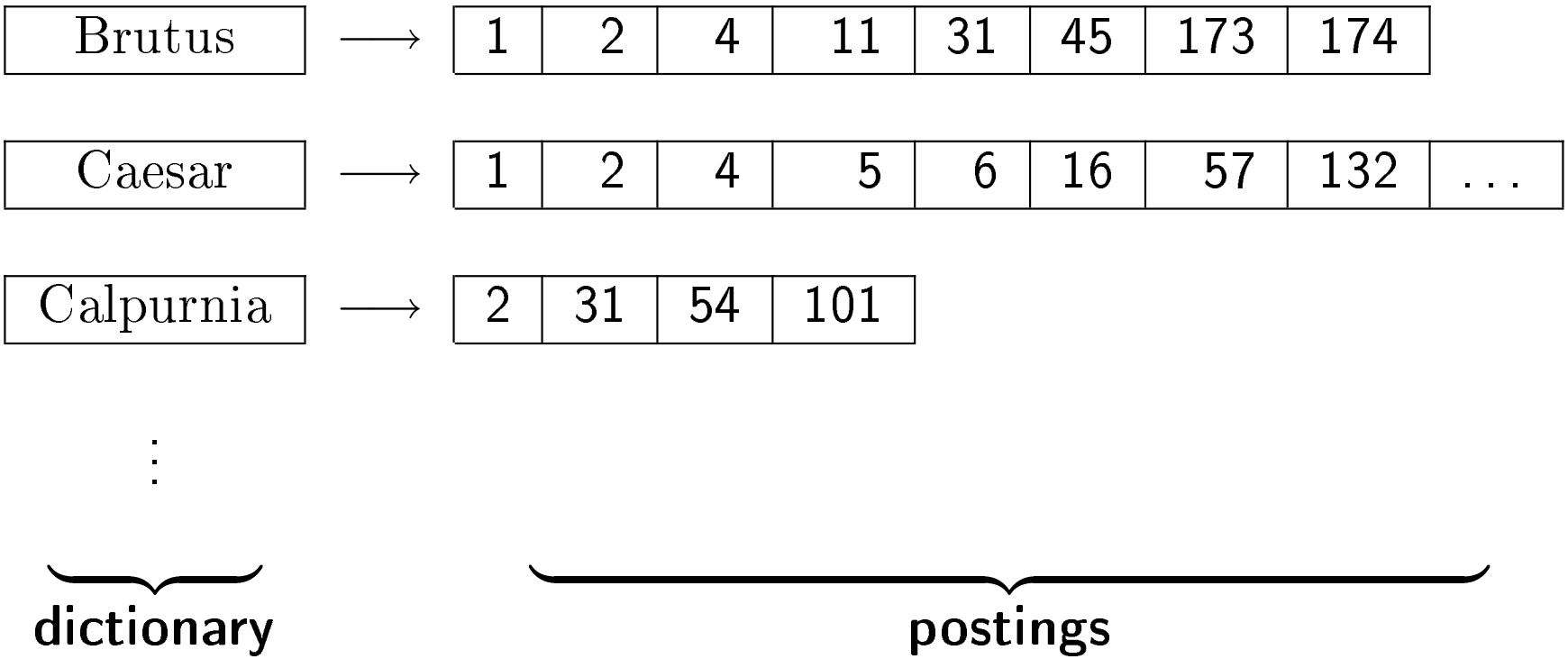
- Consider $N = 10^6$ documents, each with about 1000 tokens
- On average 6 bytes per token, including spaces and punctuation $\Rightarrow$ size of document collection is about 6 GB
- Assume there are $M = 500{,}000$ distinct terms in the collection
- (Notice that we are making a term/token distinction.)

Can't build the incidence matrix

- $M = 500{,}000 \times 10^6 =$ half a trillion 0s and 1s.
- But the matrix has no more than one billion 1s.
  - Matrix is extremely sparse.
- What is a better representations?
  - We only record the 1s.

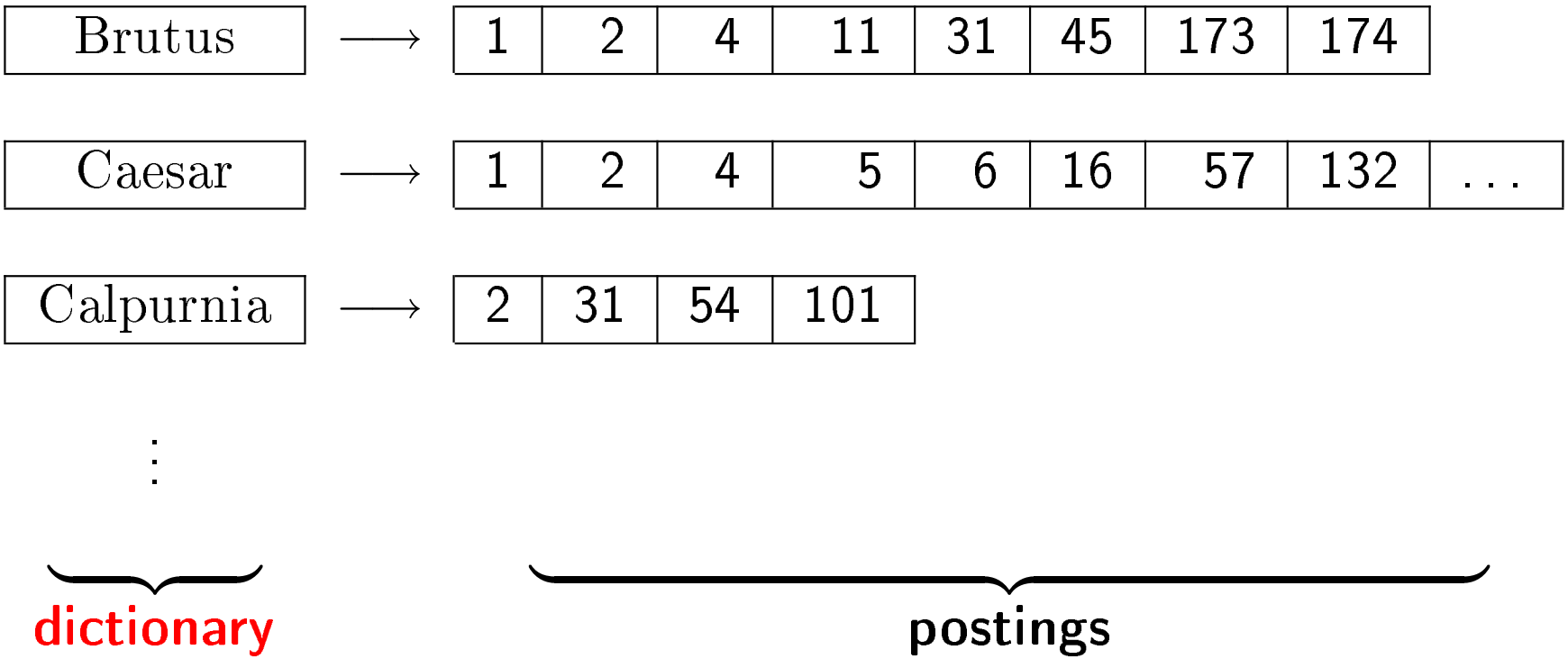
Inverted Index

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



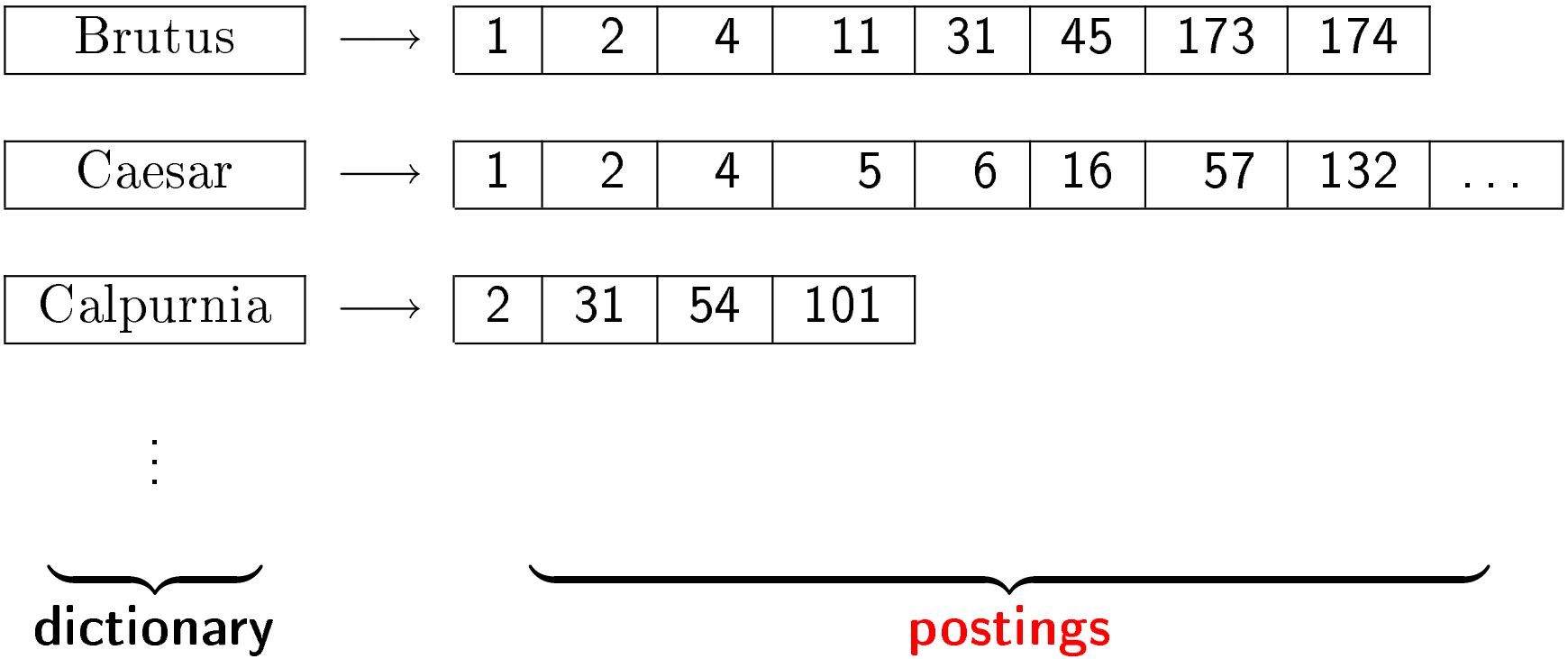
Inverted Index

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



Inverted Index

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



Inverted index construction

1. Collect the documents to be indexed:
   Friends, Romans, countrymen. | So let it be with Caesar
   ...
2. Tokenize the text, turning each document into a list of tokens:
   Friends | Romans | countrymen | So ...
3. Do linguistic preprocessing, producing a list of normalized tokens, which are the indexing terms: friend | roman | countryman | so ...
4. Index the documents that each term occurs in by creating an inverted index, consisting of a dictionary and postings.

Tokenization and preprocessing

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:

$\Longrightarrow$

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

Generate postings

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

$\Longrightarrow$

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

Sort postings

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

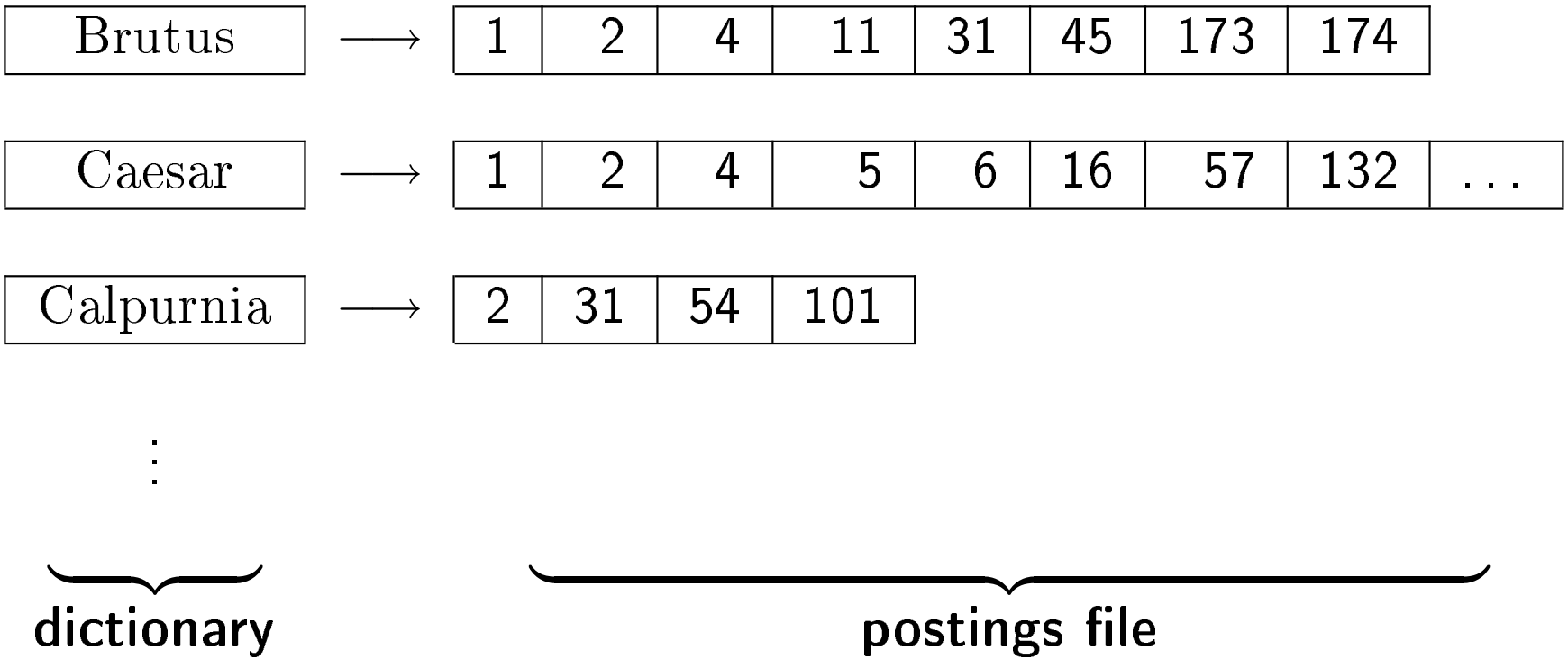
Create postings lists, determine document frequency

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

Split the result into dictionary and postings file



Outline

Simple conjunctive query (two terms)

- Consider the query: Brutus AND Calpurnia
- To find all matching documents using inverted index:

Simple conjunctive query (two terms)

- Consider the query: Brutus AND Calpurnia
- To find all matching documents using inverted index:
  1. Locate Brutus in the dictionary

Simple conjunctive query (two terms)

- Consider the query: Brutus AND Calpurnia
- To find all matching documents using inverted index:
  1. Locate Brutus in the dictionary
  2. Retrieve its postings list from the postings file

Simple conjunctive query (two terms)

- Consider the query: Brutus AND Calpurnia
- To find all matching documents using inverted index:
  1. Locate Brutus in the dictionary
  2. Retrieve its postings list from the postings file
  3. Locate Calpurnia in the dictionary

Simple conjunctive query (two terms)

- Consider the query: Brutus AND Calpurnia
- To find all matching documents using inverted index:
  1. Locate Brutus in the dictionary
  2. Retrieve its postings list from the postings file
  3. Locate Calpurnia in the dictionary
  4. Retrieve its postings list from the postings file

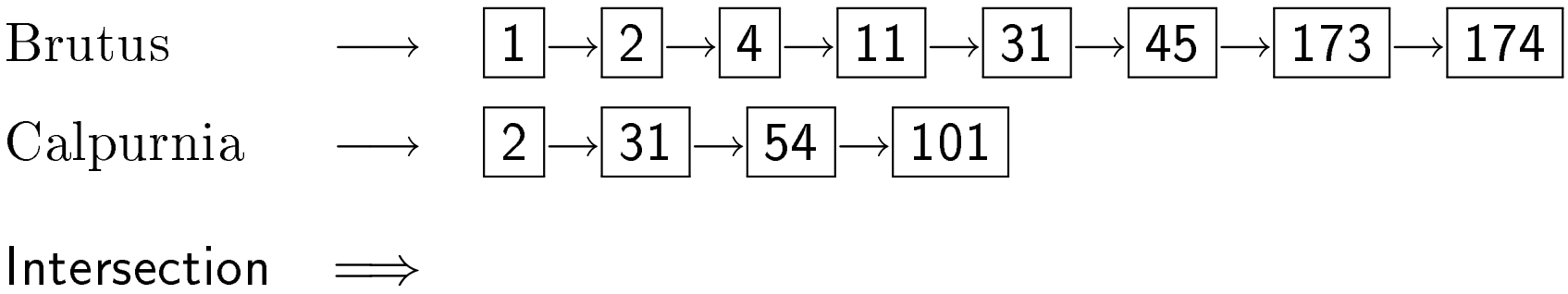
Simple conjunctive query (two terms)

- Consider the query: Brutus AND Calpurnia
- To find all matching documents using inverted index:
  1. Locate Brutus in the dictionary
  2. Retrieve its postings list from the postings file
  3. Locate Calpurnia in the dictionary
  4. Retrieve its postings list from the postings file
  5. Intersect the two postings lists

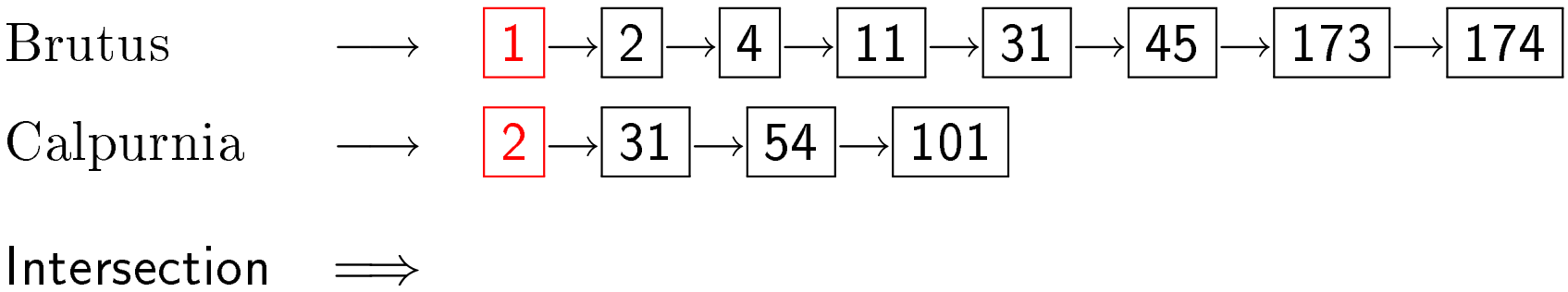
Simple conjunctive query (two terms)

- Consider the query: Brutus AND Calpurnia
- To find all matching documents using inverted index:
  1. Locate Brutus in the dictionary
  2. Retrieve its postings list from the postings file
  3. Locate Calpurnia in the dictionary
  4. Retrieve its postings list from the postings file
  5. Intersect the two postings lists
  6. Return intersection to user

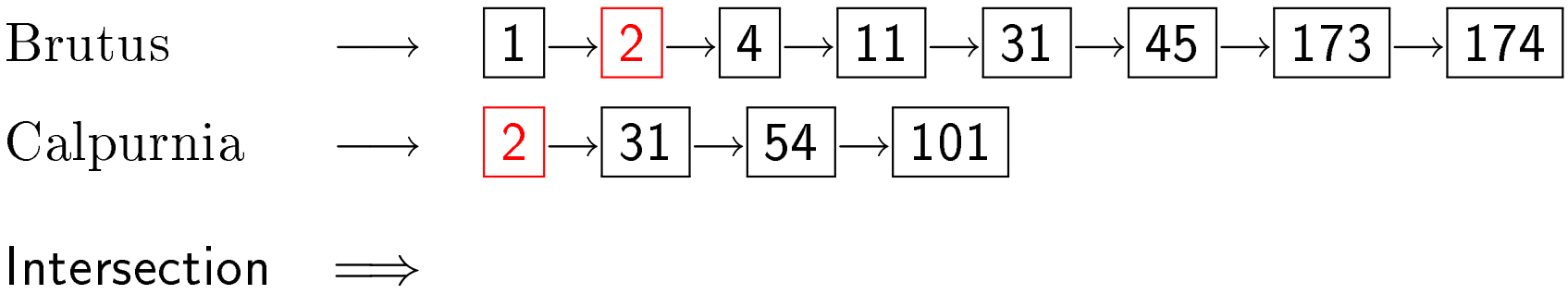
Intersecting two postings lists



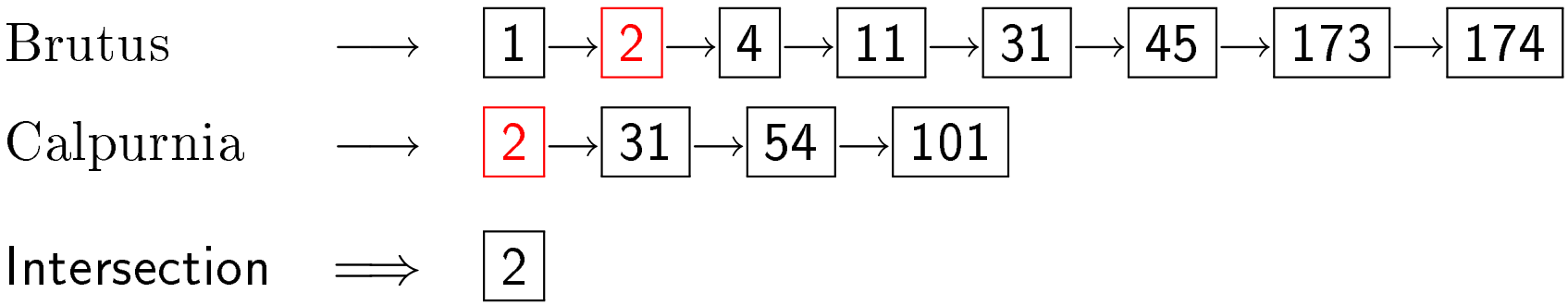
Intersecting two postings lists



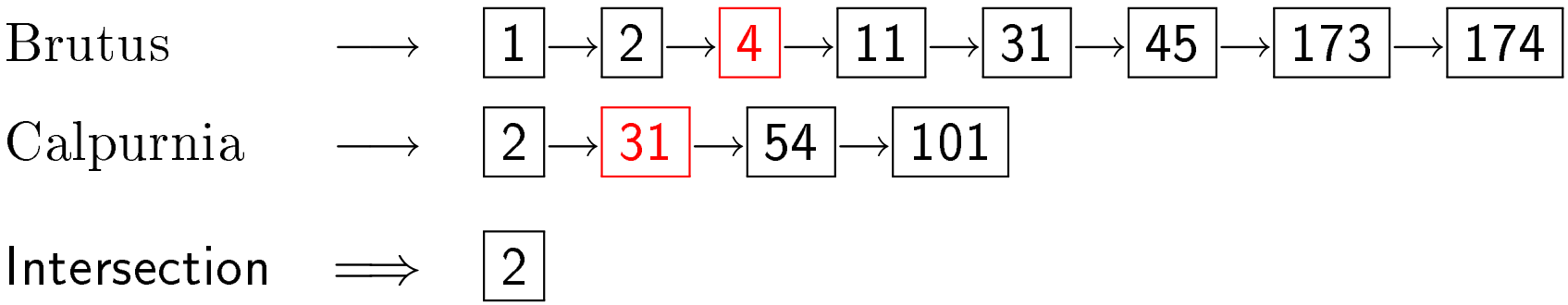
Intersecting two postings lists



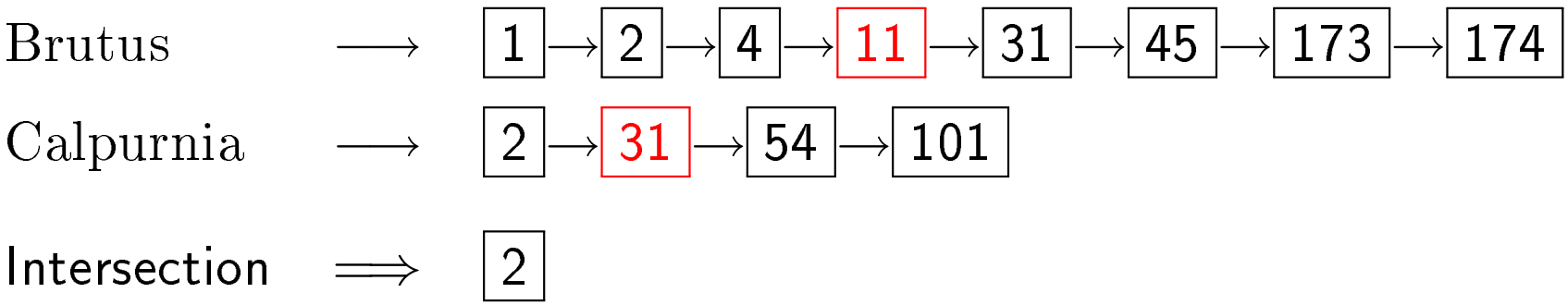
Intersecting two postings lists



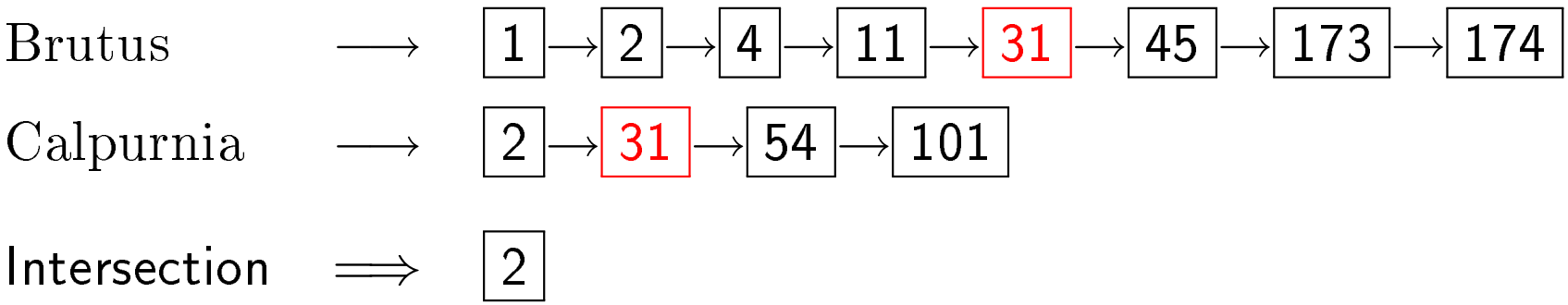
Intersecting two postings lists



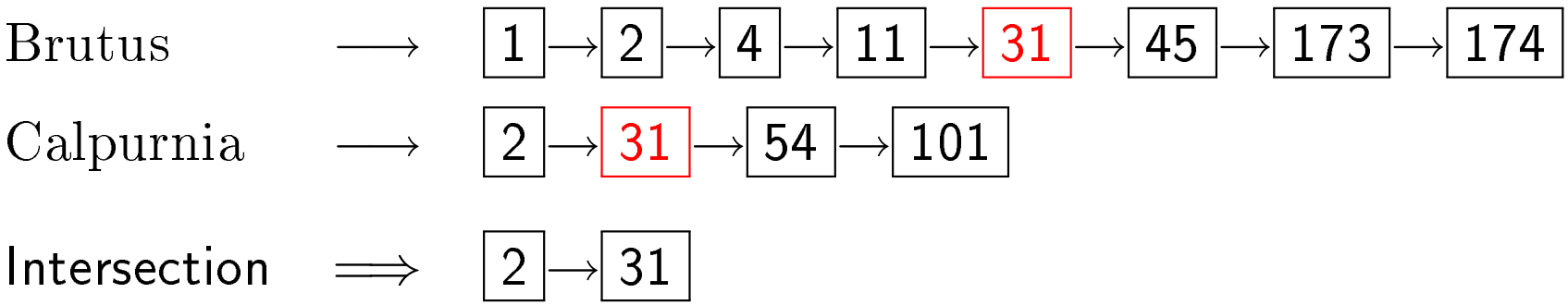
Intersecting two postings lists



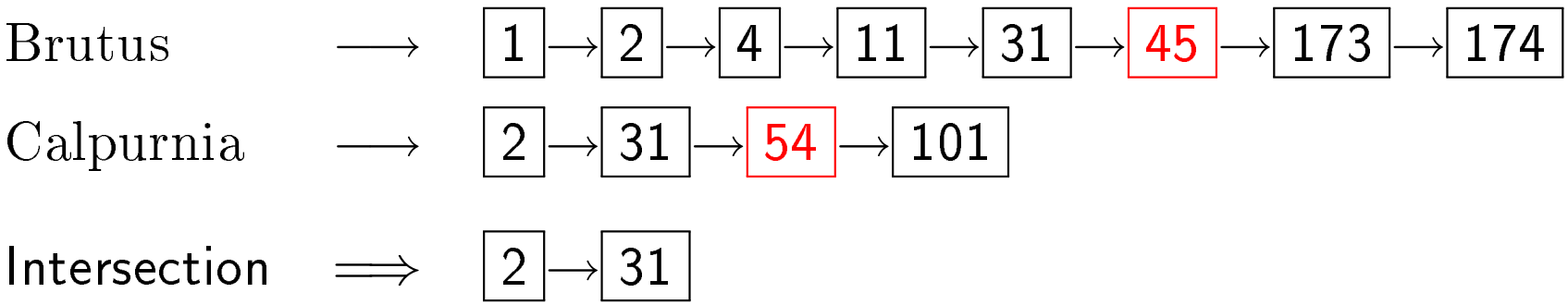
Intersecting two postings lists



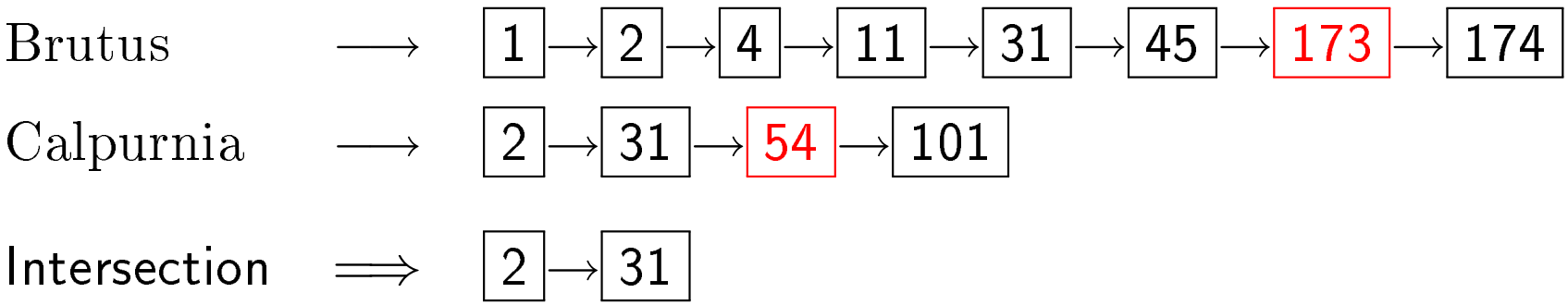
Intersecting two postings lists



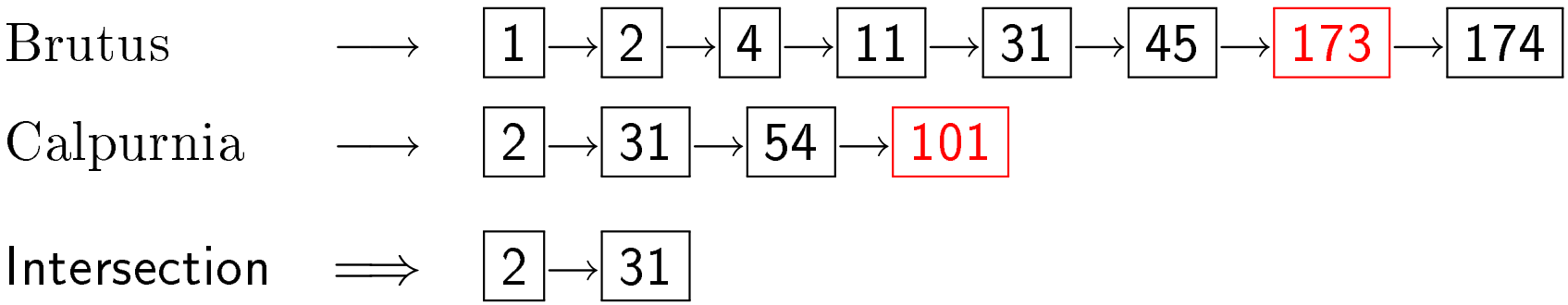
Intersecting two postings lists



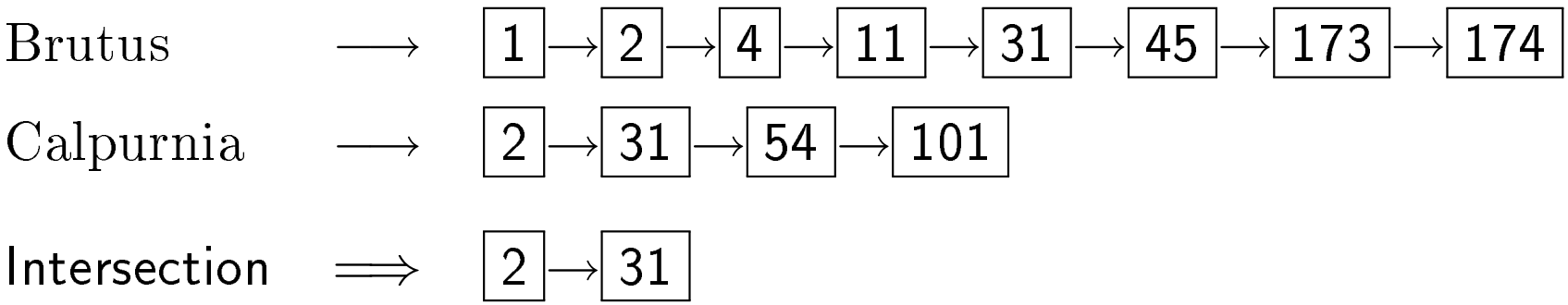
Intersecting two postings lists



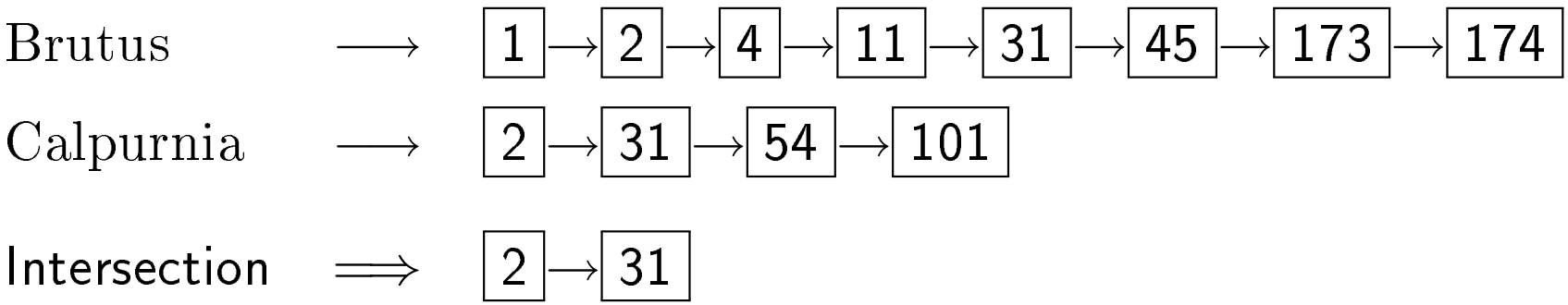
Intersecting two postings lists



Intersecting two postings lists



Intersecting two postings lists



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

Intersecting two postings lists

```
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
```

Boolean queries

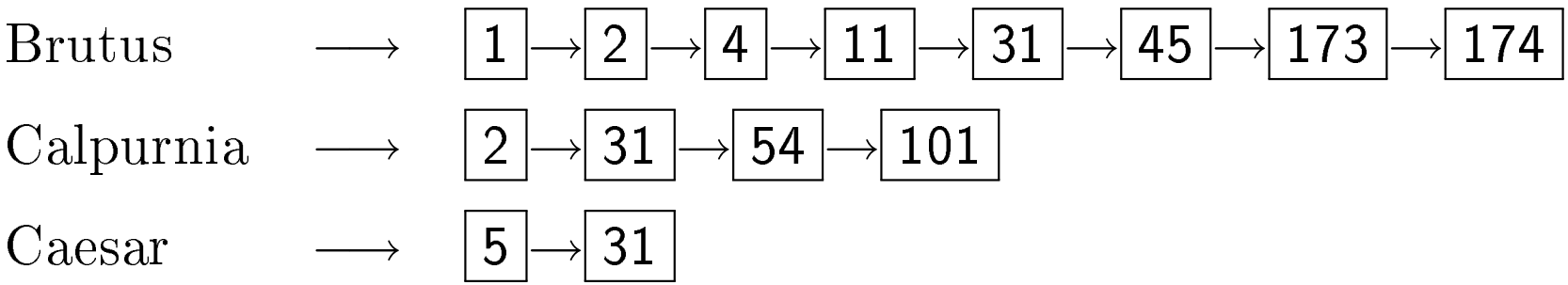
- The Boolean retrieval model can answer any query that is a Boolean expression.
  - Boolean queries are queries that use AND, OR and NOT to join query terms.
  - Views each document as a set of terms.
  - Is precise: Document matches condition or not.
- Primary commercial retrieval tool for 3 decades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

Query optimization

- What is the best order for query processing?
- 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

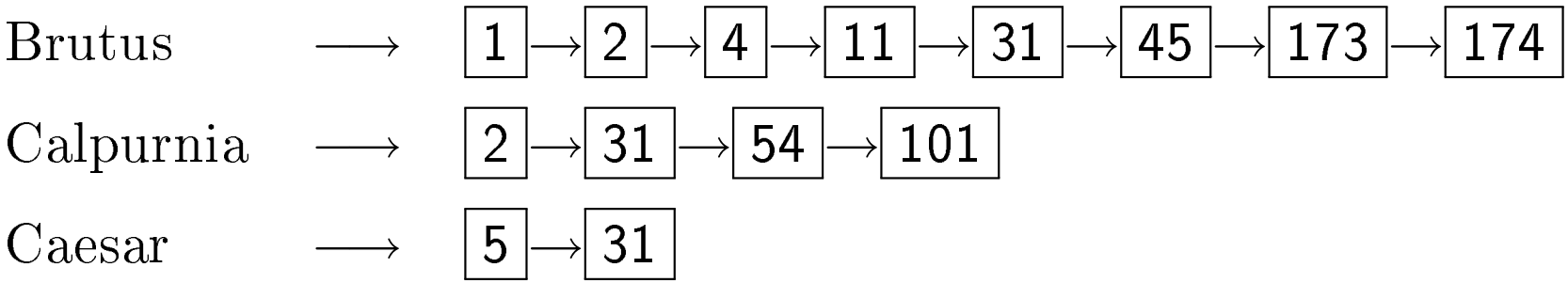
Query optimization

- Example query: Brutus AND Calpurnia AND Caesar



Query optimization

- Example query: Brutus AND Calpurnia AND Caesar
- Simple and effective optimization: Process in order of increasing frequency
- Start with the shortest postings list, then keep cutting further
- In this example, first Caesar, then Calpurnia, then Brutus



Optimized intersection algorithm for conjunctive queries

```
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
```

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

Exercise

Recommend a query processing order for: (tangerine OR trees) AND (marmalade OR skies) AND (kaleidoscope OR eyes)


H. S. Christopher Manning, P. Raghavan.
Introduction to Information Retrieval.
Cambridge, 2008.