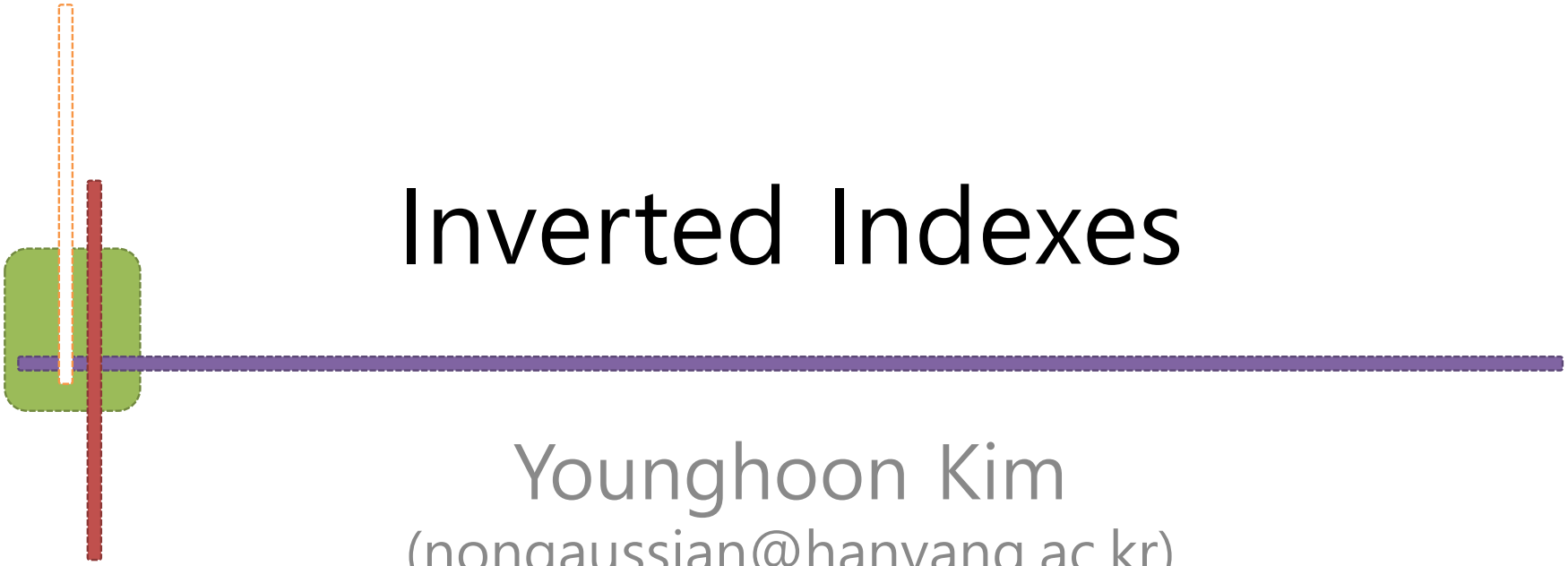


Inverted Indexes

Younghoon Kim
(nongaussian@hanyang.ac.kr)

Modified slides originally written by Jim Martin, Donald Patterson Min-Yen Kan, and Zhang & Helmer, used for the Stanford CS276 class and from the Stuttgart IIR class
https://nlp.stanford.edu/IR-book/newslides.html

TERM-DOCUMENT INCIDENCE MATRICES

Searching Unstructured Data

- 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 that not the answer?
  - Slow (for large corpora)
  - Other operations (e.g., find the word ***Romans*** near ***countrymen***) not feasible
  - Ranked retrieval (best documents to return)
    - Later lectures

Term-document Incidence Matrix

Play

	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

Keyword

1 if play contains word, 0 otherwise

Incidence Vectors and Boolean Model

So, we have a 0/1 vector for each term.

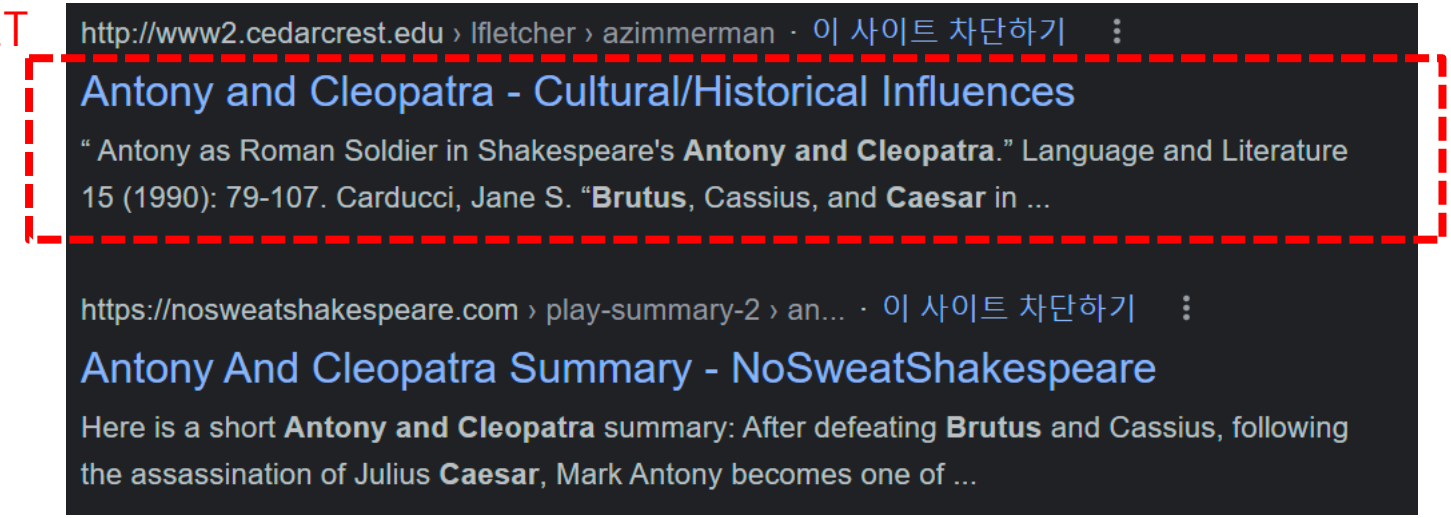
- To answer query: take the vectors for ***Brutus, Caesar*** and **not** ***Calpurnia*** (complemented) ➔ bitwise *AND*.
  - 110100 *AND*
  - 110111 *AND*
  - 101111 (= complement of 010000)
  - **100100**

	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
	1	0	0	1	0	0

Answers to Query

- Antony and Cleopatra, Act III, Scene ii

SNIPPET



Bigger collections

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

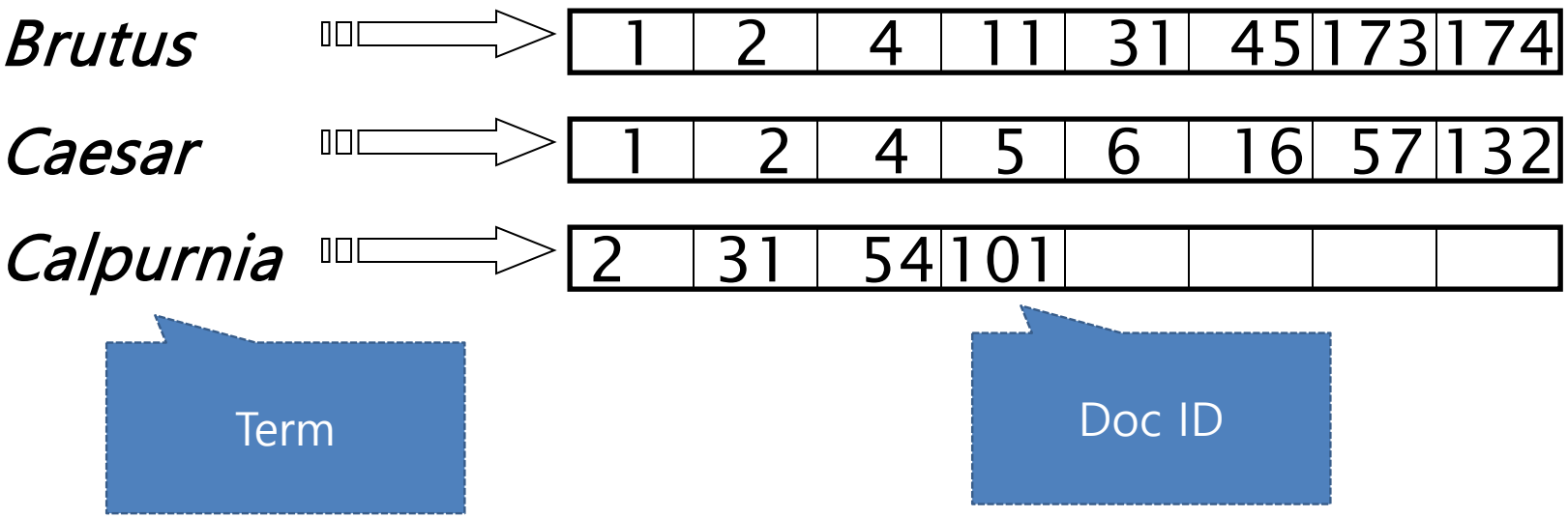
Can't build the matrix

- 500K x 1M matrix has half-a-trillion 0's and 1's
- But it has no more than one billion 1's
  - matrix is extremely sparse
- What's a better representation?
  - We only record the positions of "1"

THE INVERTED INDEX
THE KEY DATA STRUCTURE
UNDERLYING MODERN IR

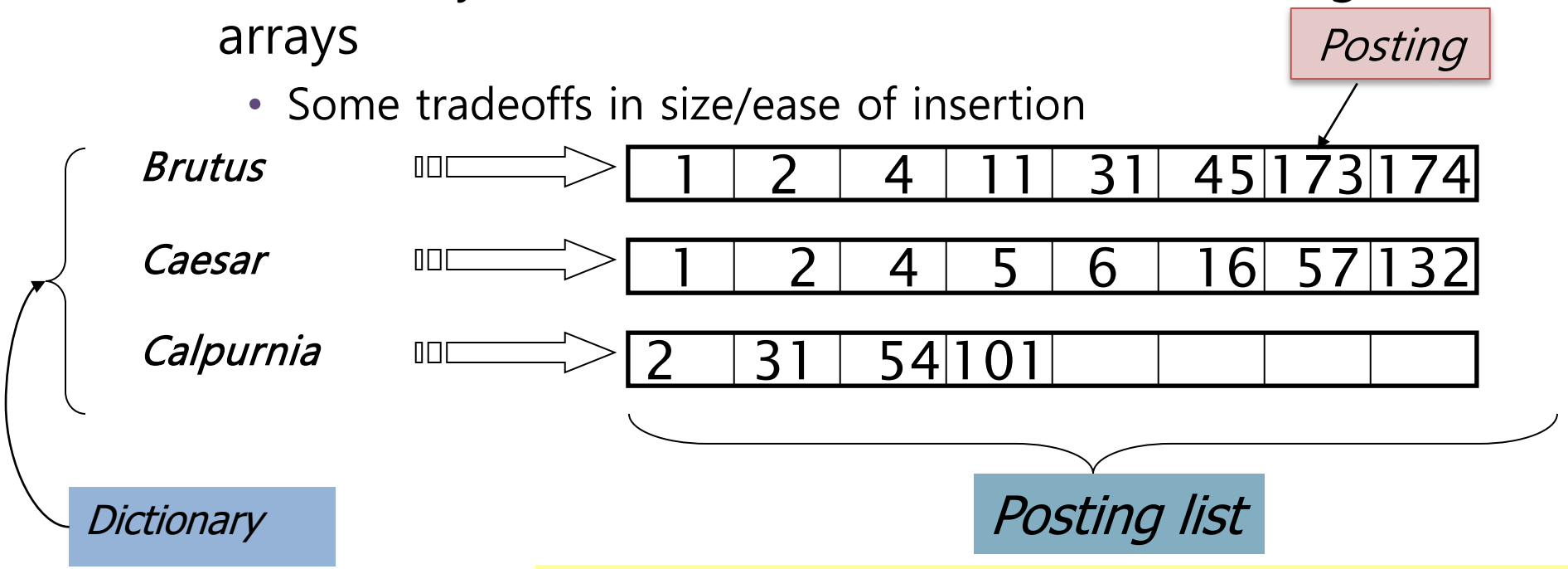
Inverted Index

- 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



Inverted index

- We need variable-size posting 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



Sorted by docID for efficient Boolean operation.

Indexer steps: Token sequence

- Sequence of (terms, doc ID, position) triplets.

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.

Term	docID	Pos
I	1	1
did	1	2
enact	1	3
julius	1	4
caesar	1	5
I	1	6
was	1	7
kill	1	8
in	1	9
the	1	10
capitol	1	11
brutus	1	12
kill	1	13
me	1	14
so	2	1
let	2	2
it	2	3
be	2	4
with	2	5
caesar	2	6
the	2	7
noble	2	8
brutus	2	9
hath	2	10
told	2	11
you	2	12
caesar	2	13
was	2	14
ambit	2	15

Indexer steps: Sort

- Sort by terms
  - And then docID

Most expensive indexing step

Term	docID	Pos
I	1	1
did	1	2
enact	1	3
julius	1	4
caesar	1	5
I	1	6
was	1	7
kill	1	8
in	1	9
the	1	10
capitol	1	11
brutus	1	12
kill	1	13
me	1	14
so	2	1
let	2	2
it	2	3
be	2	4
with	2	5
caesar	2	6
the	2	7
noble	2	8
brutus	2	9
hath	2	10
told	2	11
you	2	12
caesar	2	13
was	2	14
ambit	2	15

Sort

Term	docID	Pos
ambit	2	15
be	2	4
brutus	1	12
brutus	2	9
caesar	1	5
caesar	2	6
caesar	2	13
capitol	1	11
did	1	2
enact	1	3
hath	2	10
I	1	1
I	1	6
in	1	9
it	2	3
julius	1	4
kill	1	8
kill	1	13
let	2	2
me	1	14
noble	2	8
so	2	1
the	1	10
the	2	7
told	2	11
was	1	7
was	2	14
with	2	5
you	2	12

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	Pos
ambit	2	15
be	2	4
brutus	1	12
brutus	2	9
caesar	1	5
caesar	2	6
caesar	2	13
capitol	1	11
did	1	2
enact	1	3
hath	2	10
I	1	1
I	1	6
in	1	9
it	2	3
julius	1	4
kill	1	8
kill	1	13
let	2	2
me	1	14
noble	2	8
so	2	1
the	1	10
the	2	7
told	2	11
was	1	7
was	2	14
with	2	5
you	2	12

Dictionary	Postings	
ambit (1, 1)	2	15
be (1, 1)	2	4
brutus (2, 2)	1	12
	2	9
caesar (2, 3)	1	5
	2	6
	2	13
capitol (1, 1)	1	11
did (1, 1)	1	2
enact (1, 1)	1	3
hath (1, 1)	2	10
I (1, 1)	1	1
	1	6
in (1, 1)	1	9
it (1, 1)	2	3
julius (1, 1)	1	4
kill (1, 2)	1	8
	1	13
let (1, 1)	2	2
me (1, 1)	1	14
noble (1, 1)	2	8
so (1, 1)	2	1
the (2, 2)	1	10
	2	7
told (1, 1)	2	11
was (2, 2)	1	7
	2	14
with (1, 1)	2	5
you (1, 1)	2	12

Where do we pay in storage?

Term, document frequency and postings size

Lists of (docID, pos)'s

Dictionary	Postings	
ambit (1, 1)	2	15
be (1, 1)	2	4
brutus (2, 2)	1	12
	2	9
caesar (2, 3)	1	5
	2	6
	2	13
capitol (1, 1)	1	11
did (1, 1)	1	2
enact (1, 1)	1	3
hath (1, 1)	2	10
I (1, 1)	1	1
	1	6
in (1, 1)	1	9
it (1, 1)	2	3
julius (1, 1)	1	4
kill (1, 2)	1	8
	1	13
let (1, 1)	2	2
me (1, 1)	1	14
noble (1, 1)	2	8
so (1, 1)	2	1
the (2, 2)	1	10
	2	7
told (1, 1)	2	11
was (2, 2)	1	7
	2	14
with (1, 1)	2	5
you (1, 1)	2	12

IR system implementation

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