Lecture 5.2:
Inverted Indexes

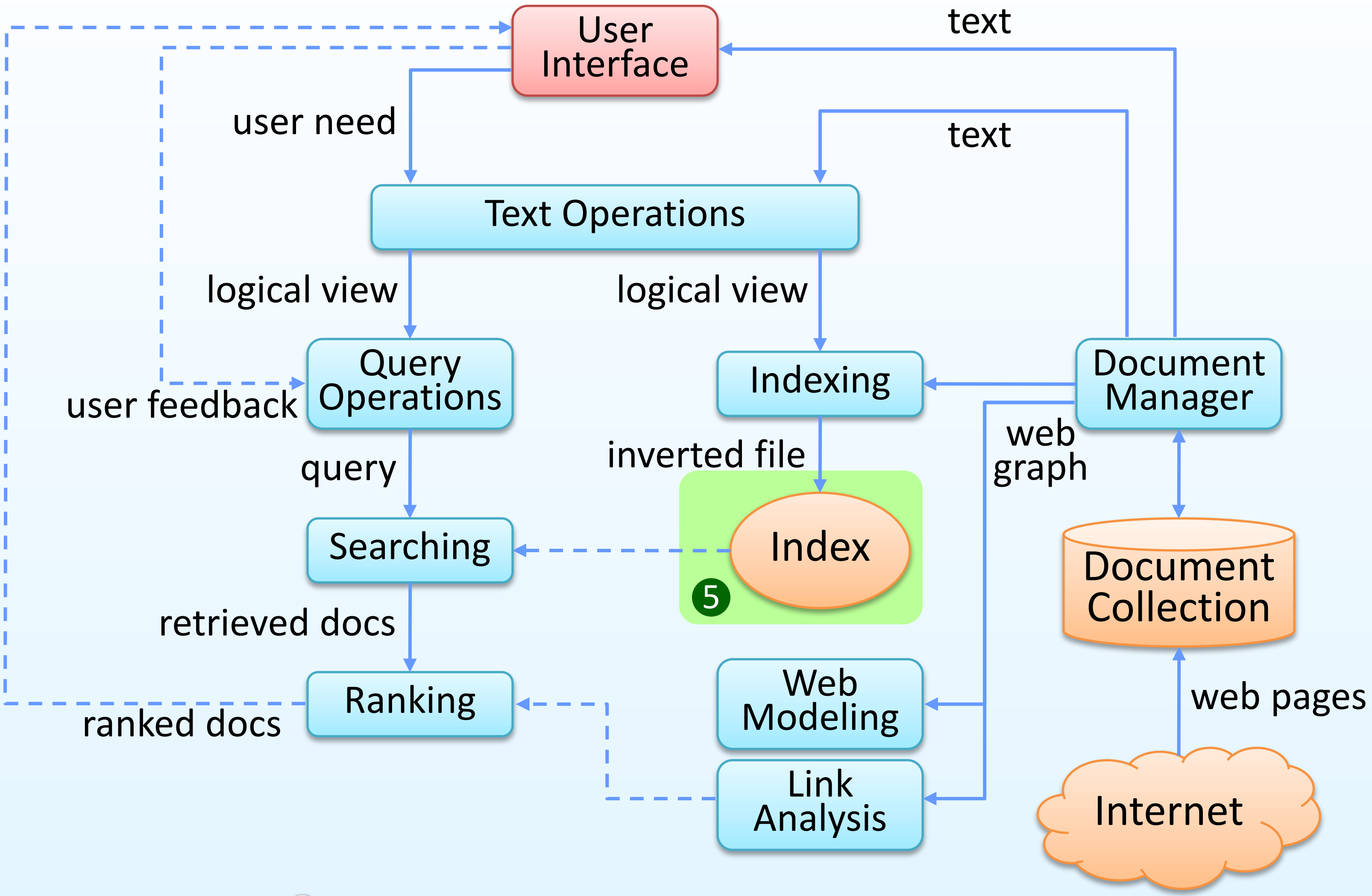
01204553 Web Information Retrieval and Mining

Department of Computer Engineering
Faculty of Engineering, Kasetsart University
Bangkok, Thailand.

Department of Computer Engineering Kasetsart University

MIKE LABORATORY

Review: Search Engine Architecture



Introduction

- Although efficiency might seem a secondary issue compared to effectiveness, it should not be neglected in the design of an IR system.

- Efficiency in IR systems
  - To process indexing and searching with minimal requirements of computational resources

- As we move to large-scale application, efficiency becomes more and more important
  - For example, web search engine with petabytes of indexed data and billion of queries per second

Introduction

- **Index**: a data structure built from text to speed up the searches

- In the context of an IR system, the efficiency can be measured by
  - Indexing time: time needed to build the index
  - Indexing space: space used during the generation of the index
  - Index storage: space required to store the index
  - Query latency: time interval between the arrive of the query and the generation of the answer
  - Query throughput: average number of queries processed per second

Inverted Indexes

Review: Term-Document Matrix

d_1
To do is to be.
To be is to do.

d_2
To be or not to be.
I am what I am.

d_3
I think therefore I am.
Do be do be do.

d_4
Do do do, da da da.
Let it be, let it be.

Vocabulary	n_i	d_1	d_2	d_3	d_4
to	2	4	2	-	-
do	3	2	-	3	3
is	1	2	-	-	-
be	4	2	2	2	2
or	1	-	1	-	-
not	1	-	1	-	-
I	2	-	2	2	-
am	2	-	2	1	-
what	1	-	1	-	-
think	1	-	-	1	-
therefore	1	-	-	1	-
da	1	-	-	-	3
let	1	-	-	-	2
it	1	-	-	-	2

Problems of Term-Document Matrix

Problem: It requires too much space (sparse matrix).

Solution: Represent it as a term-document list.

Problem: It requires sequential comparing between each document and a user's query, $sim(d_j, q)$.

Solution: Represent it as a set of occurrence lists, called **inverted index**.

Basic Concepts

- Inverted index
  - A word-oriented mechanism for indexing a text collection to speed up the searching task

- The inverted index structure is composed of two elements:
  - Vocabulary—the set of all different words in the text
  - Occurrences—for each word in the vocabulary, occurrences refer to documents containing that word

Basic Inverted Index

d_1 To do is to be.
To be is to do.

d_2 To be or not to be.
I am what I am.

d_3 I think therefore I am.
Do be do be do.

d_4 Do do do, da da da.
Let it be, let it be.

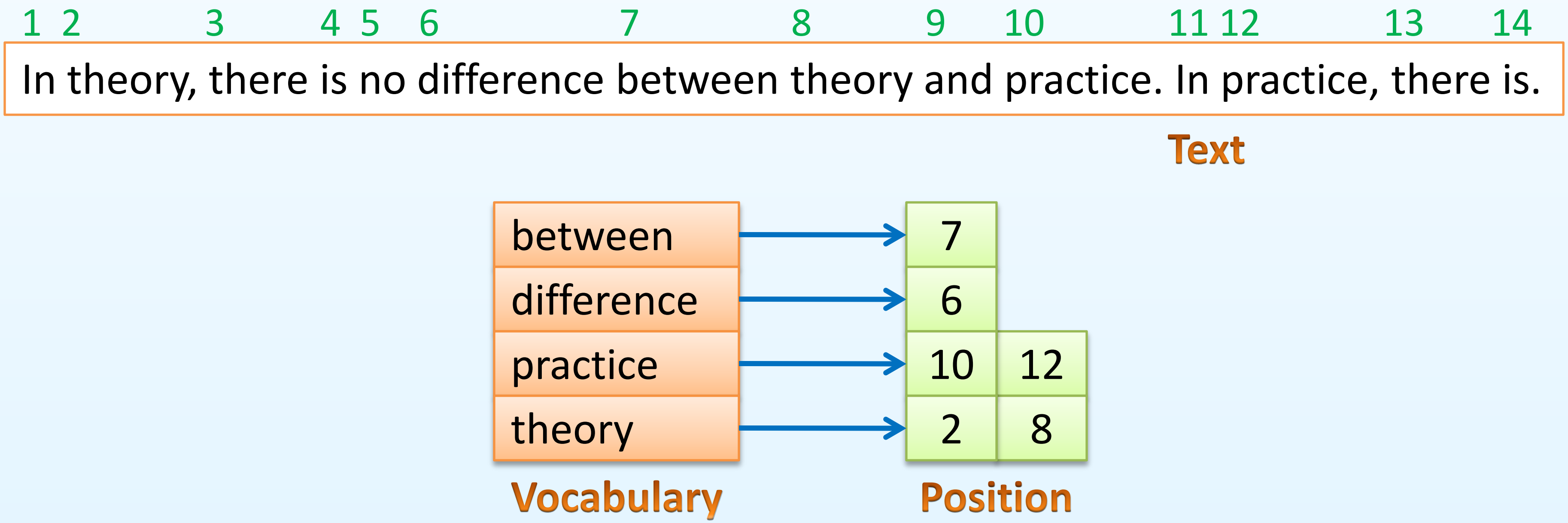
Vocabulary	n_i	Occurrences
to	2	[1,4],[2,2]
do	3	[1,2],[3,3],[4,3]
is	1	[1,2]
be	4	[1,2],[2,2],[3,2],[4,2]
or	1	[2,1]
not	1	[2,1]
I	2	[2,2],[3,2]
am	2	[2,2],[3,1]
what	1	[2,1]
think	1	[3,1]
therefore	1	[3,1]
da	1	[4,3]
let	1	[4,2]
it	1	[4,2]

Full Inverted Indexes

Full Inverted Index

- The basic index is not suitable for answering phrase or proximity queries.

- Hence, we need to add the positions of each word in each document to the index (full inverted index)



1 2 3 4 5 6 7 8 9 10 11 12 13 14

In theory, there is no difference between theory and practice. In practice, there is.

Text

Vocabulary	Position	
between	7	
difference	6	
practice	10	12
theory	2	8

Full Inverted Index

d_1

To do is to be.
To be is to do.

d_2

To be or not to be.
I am what I am.

d_3

I think therefore I am.
Do be do be do.

d_4

Do do do, da da da.
Let it be, let it be.

Vocabulary	n_i	Occurrences with position
to	2	[1,4,[1,4,6,9]],[2,2,[1,5]]
do	3	[1,2,[2,10]],[3,3,[6,8,10]],[4,3,[1,2,3]]
is	1	[1,2,[3,8]]
be	4	[1,2,[5,7]],[2,2,[2,6]],[3,2,[7,9]],[4,2,[9,12]]
or	1	[2,1,[3]]
not	1	[2,1,[4]]
I	2	[2,2,[7,10]],[3,2,[1,4]]
am	2	[2,2,[8,11]],[3,1,[5]]
what	1	[2,1,[9]]
think	1	[3,1,[2]]
therefore	1	[3,1,[3]]
da	1	[4,3,[4,5,6]]
let	1	[4,2,[7,10]]
it	1	[4,2,[8,11]]

Any Question?