BITS Pilani
Pilani Campus

Information Retrieval

Abhishek
January 2020

innovate achieve lead

BITS Pilani
Pilani Campus

CS F469, Information Retrieval
Lecture No. 2

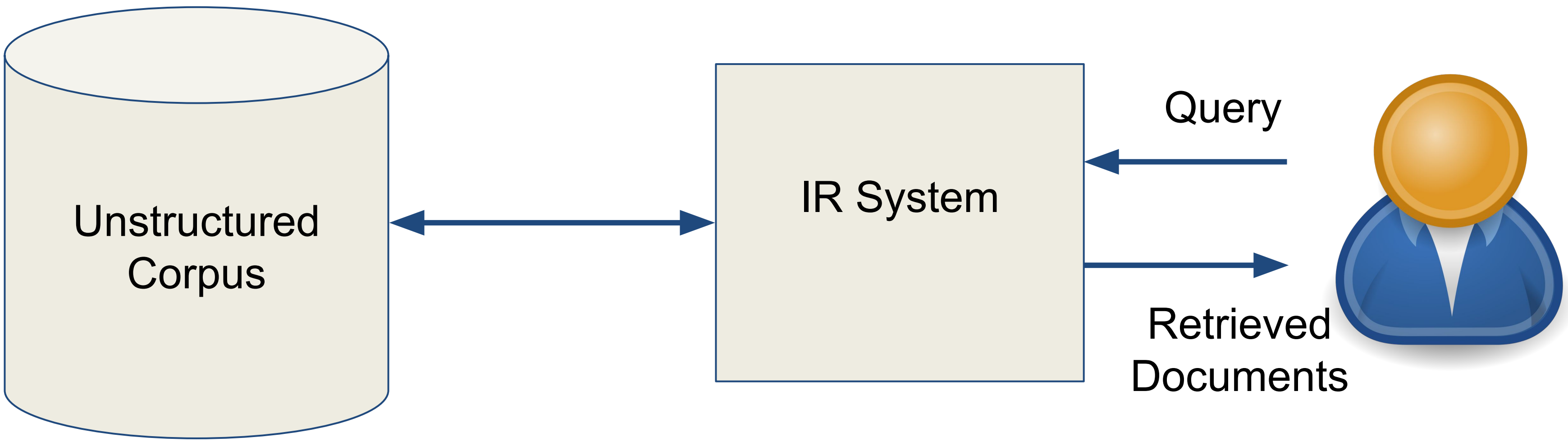
Recap of Lecture 1

- What is Information Retrieval (IR)?
- Why do we need IR?
- Why IR is important?
- Course overview

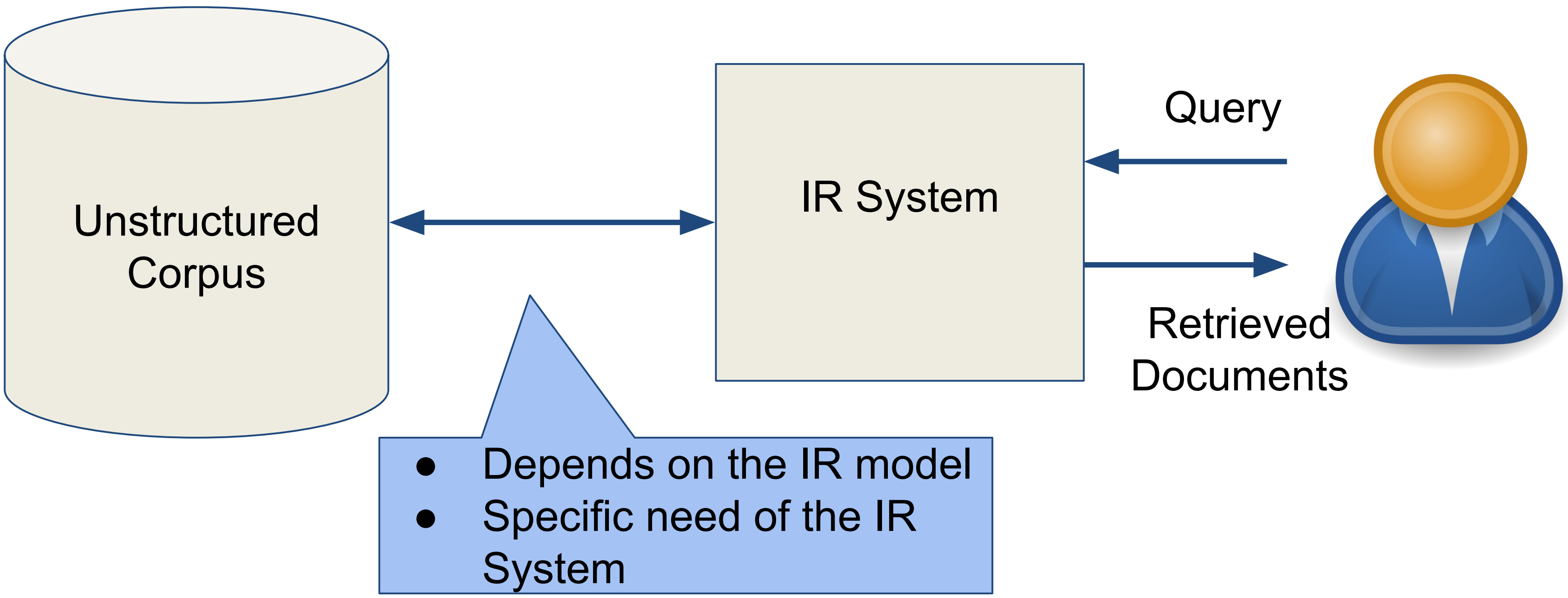
Today’s Lecture

- A simple IR task
- Boolean Retrieval models
- Indexing

Overview of IR system



Overview of IR system



Key terminologies

- **Document:** A unit that we have decided to build a retrieval system. It can be web pages, book chapters, research papers etc.
- **Collection/Corpus:** A group of documents over which we perform retrieval.

Key terminologies

- **Information Need:** A topic about which a user wants to know more.
- **Query:** Some words or phrases that the user writes in the computer in an attempt to communicate the information need.

Boolean Retrieval Model

An IR model, in which user can pose queries as boolean expressions.

Example of query:

Virat AND Anushka

An Example IR problem

Documents: D_1, D_2, …., D_N

Average words per document: A_{wpd}

Unique words: M

Model: Boolean retrieval model.

Naive solution: Linear scan

Linear scan: For every query, scan the corpus and find relevant documents.

Major limitation:

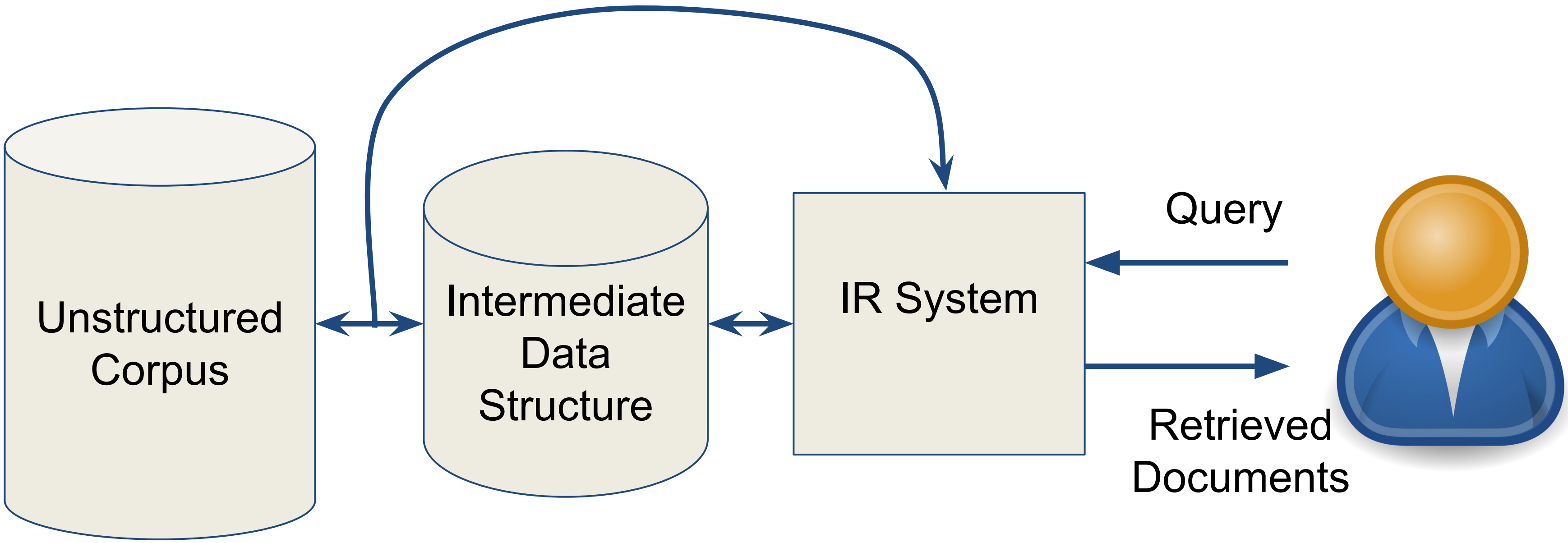
- For every query, need to process whole corpus, i.e., $N * A_{wpd}$ words.
- If N is large (in range of million and more documents), not feasible to implement a practical IR system on a decent computer.

Naive solution: Linear scan

Advantage: Linear scan is the only solution if we only have access to the corpus with no additional memory or storage space.

Better solutions require an intermediate data structure.

Overview of IR system



Solution 2: Incidence Matrix

	Document 1	Document 2	Document 3	Document 4	...	Document N
Word 1	1	0	1	0		1
Word 2	0	1	0	0		0
Word 3	0	0	0	0		0
Word 4	0	1	1	0		0
Word 5	1	0	1	1		1
Word 6	1	1	0	1		1
:						
Word M	0	0	1	0		1

Solution 2: Incidence Matrix

Advantage: For every query, the system needs to access few rows and perform boolean operations on the rows.

Solution 2: Incidence Matrix

Eg. query: **word 1** AND **word 6**

Word 1	1	0	1	0	**...**	1

AND

Word 6	1	1	0	1	**...**	1

=

Result	1	0	0	0	**...**	1

Solution 2: Incidence Matrix

Eg. query: **word 1** AND **word 6**

Word 1	1	0	1	0	...	1

AND

Word 6	1	1	0	1	...	1

=

Result	1	0	0	0	...	1

For every query: N * (query words - 1) AND operations

Solution 2: Incidence Matrix

Limitations:

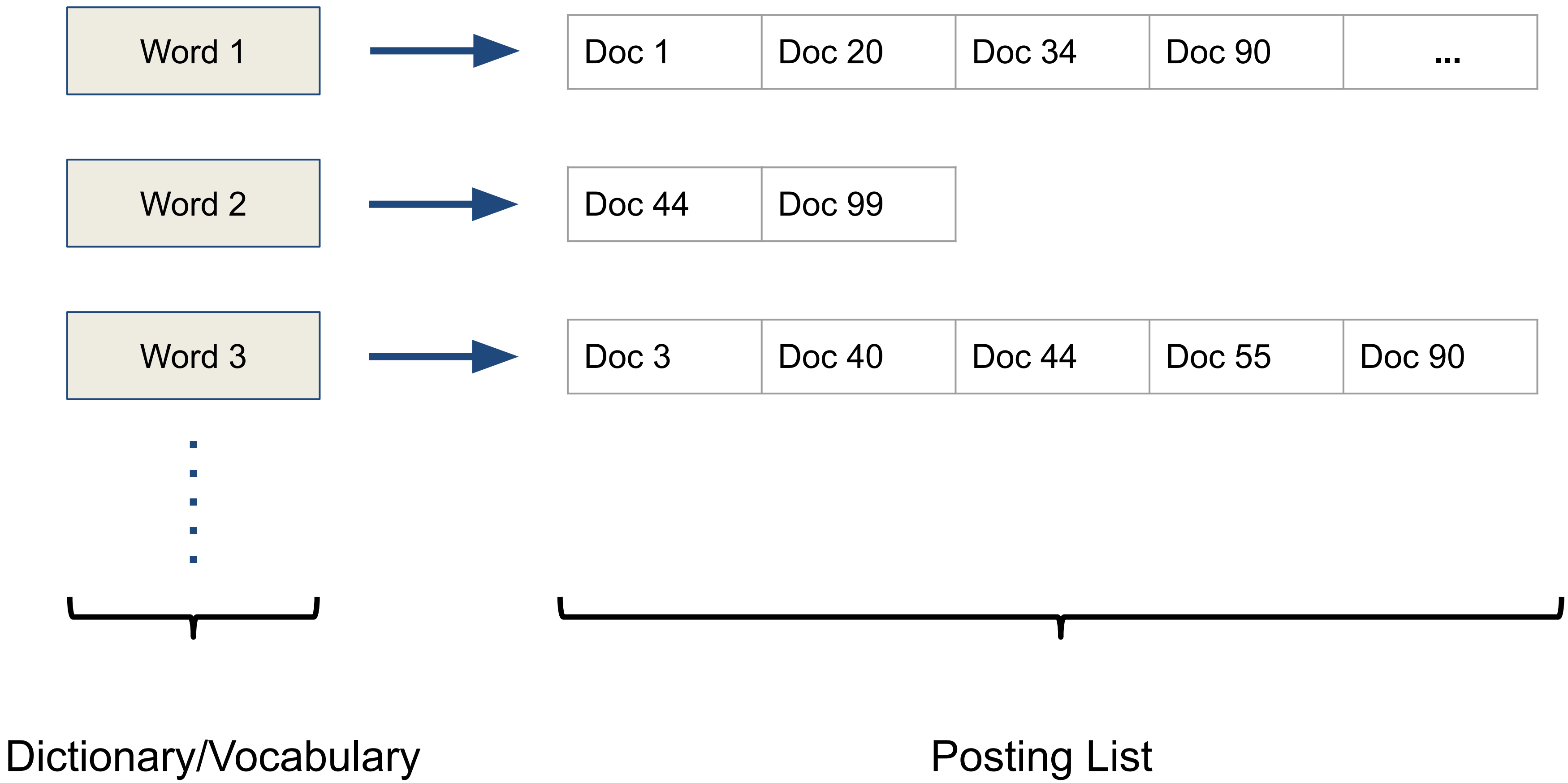
The matrix size will be huge for general corpus.

- For every new document, the matrix size will increase by at least M.

Observation in text corpus: Other than few common words, every other words does not appears in every document.

Solution 3: Inverted Index

Solution 3: Inverted Index



Solution 3: Inverted Index

Advantage: For every query, the system needs to access few elements of dictionary and perform intersection of the posting lists.

Solution 3: Inverted Index

Eg. query: **word 1** AND **word 3**

Word 1	→	Doc 1	Doc 20	Doc 34	Doc 90	...

intersection

Word 3	→	Doc 3	Doc 40	Doc 44	Doc 55	Doc 90

=

Result	→	Doc 90

Solution 3: Inverted Index

Eg. query: **word 1** AND **word 3**

Word 1 → | Doc 1 | Doc 20 | Doc 34 | Doc 90 | ... |

intersection

Word 3 → | Doc 3 | Doc 40 | Doc 44 | Doc 55 | Doc 90 |

=

Result → | Doc 90 |

If the posting lists are sorted, then for every query:
O(N) * (query words) operations to find intersection.

Posting List Intersection Algo.

```
INTERSECT(p1, p2)
  answer ← ⟨ ⟩
  while p1 ≠ NIL and p2 ≠ NIL
  do if docID(p1) = docID(p2)
       then ADD(answer, docID(p1))
            p1 ← next(p1)
            p2 ← next(p2)
       else if docID(p1) < docID(p2)
              then p1 ← next(p1)
              else p2 ← next(p2)
  return answer
```

Source: Figure 1.6, [Introduction to Information Retrieval]

Solution 3: Inverted Index

Observations:

- The query processing time for inverted index is asymptotically same as that of incidence matrix.
- However, in practice it takes less time because not every query will have the posting list size close to N.
- Query optimization can be done to reduce time further.

Solution 3: Inverted Index

Query Optimization:

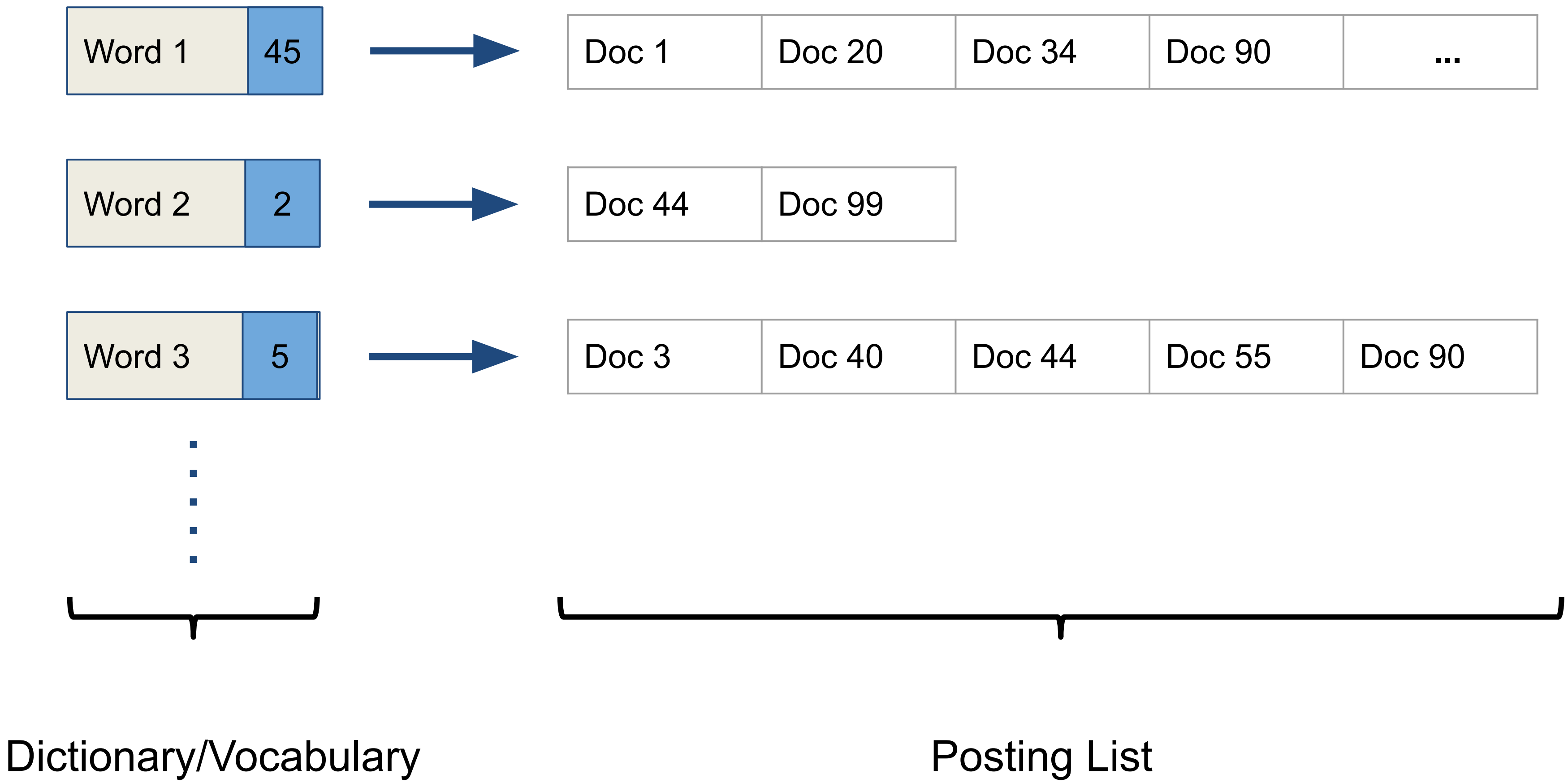
Eg. Query: **word 1** AND **word 2** AND **word 3**

Let posting list size for word 1, word 2 and word 3 are 100, 50 and 10, respectively.

In what order we should process this query?

1. **(word 1** AND **word 2)** AND **word 3**
2. **word 1** AND **(word 2** AND **word 3)**

Inverted Index with Term Frequency



Building an Inverted Index

Step 1: Collect the documents to be indexed.

Example with two documents.

Doc 1:

A quick brown fox jumps over a lazy dog.

Doc 2:

The quick sly fox jumped over the lazy brown dog.

Building an Inverted Index

Step 2: Tokenize the documents into list of tokens.

Doc 1:

A	quick	brown	fox	jumps	over	a	lazy	dog	.

Doc 2:

The	quick	sly	fox	jumped	over	the	lazy	brown	dog	.

Building an Inverted Index

Step 3: Do some linguistic preprocessing, eg. lowercase

Doc 1:

a	quick	brown	fox	jumps	over	a	lazy	dog

Doc 2:

the	quick	sly	fox	jumped	over	the	lazy	brown	dog

Building an Inverted Index

Step 4: Build the inverted index considering the tokens as terms.

a	1	the	2
quick	1	quick	2
brown	1	sly	2
fox	1	fox	2
jumps	1	jumped	2
over	1	over	2
a	1	the	2
lazy	1	lazy	2
dog	1	brown	2
		dog	2

Building an Inverted Index

Step 4: Build the inverted index considering the tokens as terms.

a	1
brown	1, 2
dog	1, 2
fox	1, 2
jumped	1
jumps	1
lazy	1, 2
over	1, 2
quick	1, 2
sly	2
the	2

Boolean Retrieval system using Google

Exercise: What is the syntax of AND, OR and NOT operators in google search?

Reference

https://nlp.stanford.edu/IR-book/

Chapter 1

Thank You!