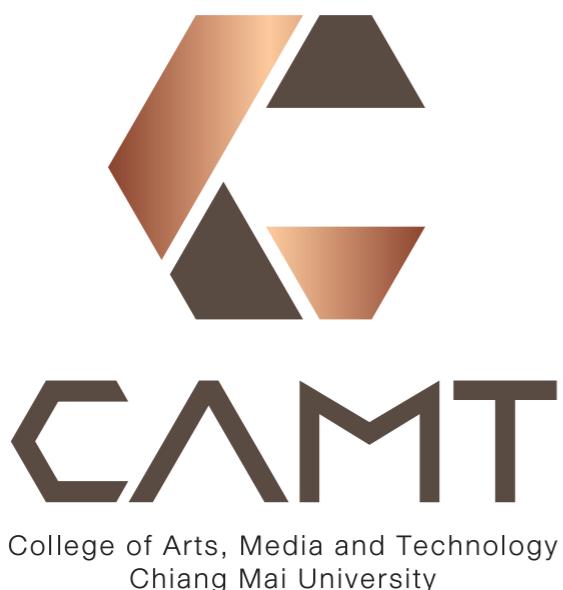


SE 481 Introduction to Information Retrieval

(IR for SE)

Module #1 — Overview



Passakorn Phannachitta, D.Eng.

passakorn.p@cmu.ac.th

College of Arts, Media and Technology
Chiang Mai University, Chiangmai, Thailand

Information Retrieval (IR)

- **Information retrieval** is the science of searching for information in a document, searching for documents themselves, and also searching for the **metadata** that describes data, and for databases of texts, images or sounds.

— Wikipedia

Shorter version

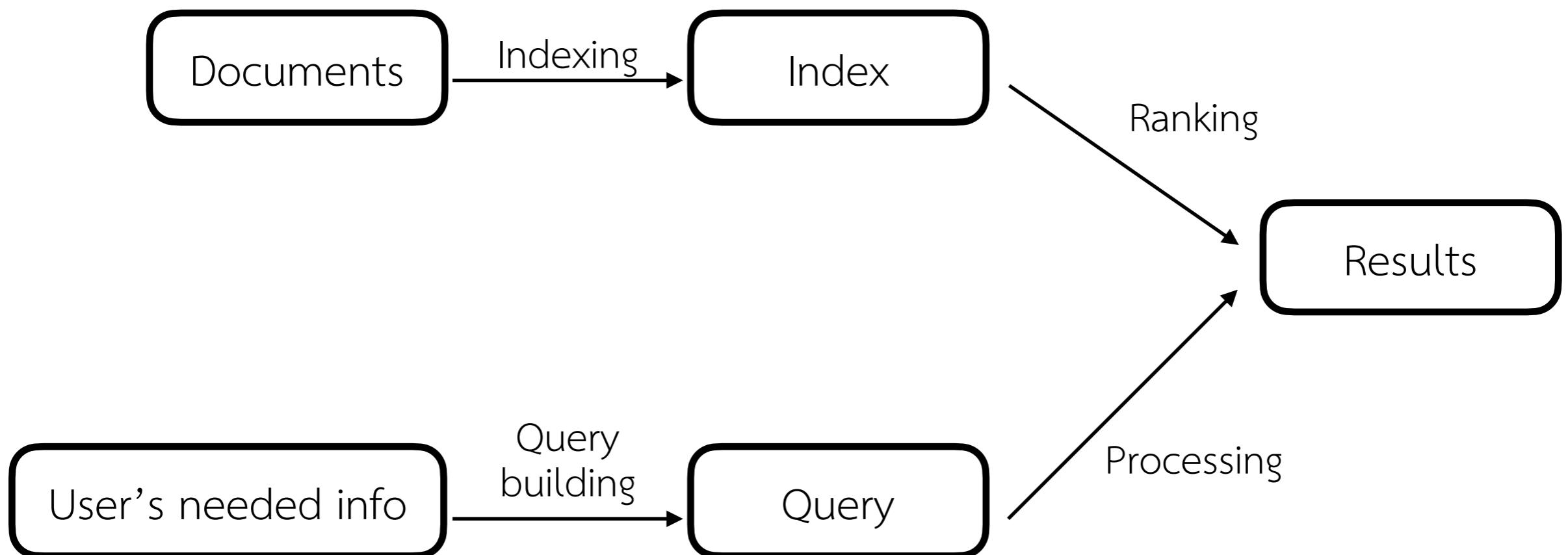
- IR is to connect between:
 - Users —> who want information
 - Documents —> provide information

Shorter version

- IR is to connect between:
 - Users —> who want information
 - Documents —> provide information
- The two ends cannot establish a direct communication
- Best is to guess
- However, the information source used for guessing is **limited**

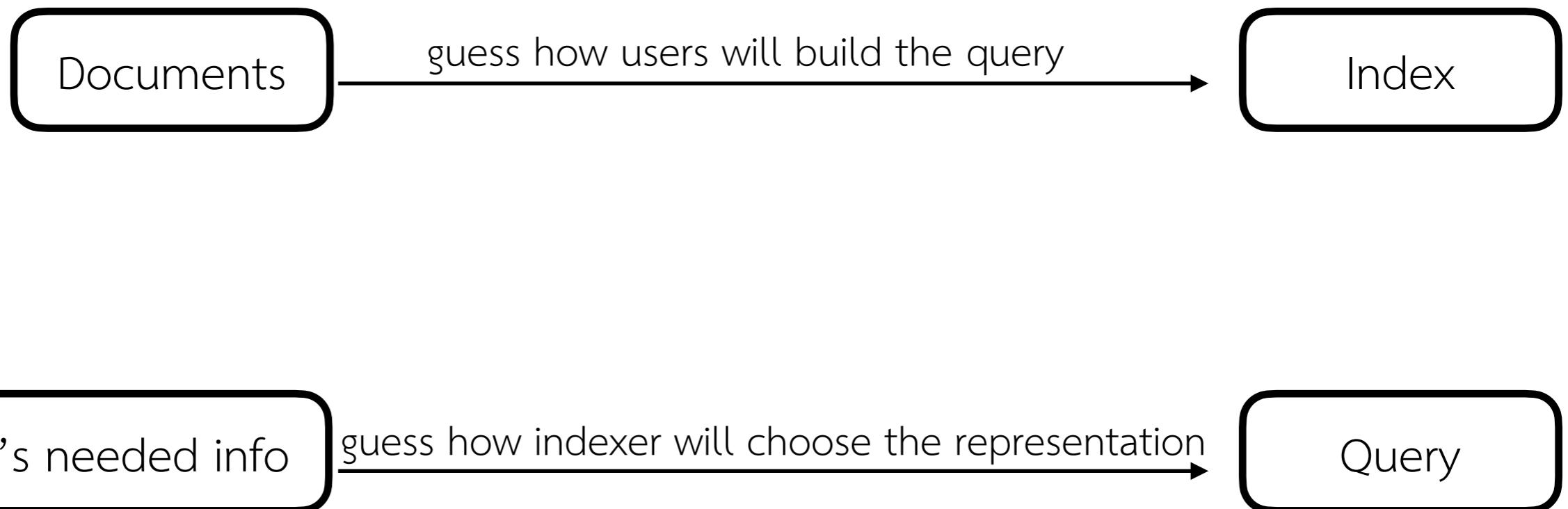
Users <> Documents

- IR



The connecting point

- IR



Another view

- IR is a search for similarity:
 - Documents and queries
 - Documents in collection
 - Users

Basic assumption of IR

- Information is stored in a set of documents
- An IR system retrieves documents with information that is believed to be most relevant to the **user's information need** and helps the user complete a **task**
- Data are born **unstructured**

Finding an answer for any question, e.g.,

Which character has the most deaths to his/her name?

Asked 7 years, 6 months ago Active 2 years, 10 months ago Viewed 134k times

15

In the Dragon Ball series, we see tons of people dying and being revived by the dragon balls. This triggered my curiosity to know which character has the most deaths to his/her name in the whole Dragon Ball series. Who is it, and how many times did he/she die?

dragon-ball-series



Ref: <https://anime.stackexchange.com/questions/8851/which-character-has-the-most-deaths-to-his-her-name>

Ref: https://en.wikipedia.org/wiki/List_of_Dragon_Ball_characters

For a question sounded more SE

↑ Posted by u/MossRock42 7 years ago ━

1 **What is the best programming language for career development as a software engineer?**

This is a question that I've been wondering for a while now. My current job is mostly doing C#/MVC Web Applications. But I am considering trying for a better job elsewhere. I see a lot of job postings asking for things like Ruby, Python, Java, NodeJS, etc. So if you had to pick one which one would it be?

14 Comments Award Share Save Hide Report 57% Upvoted

Ref: https://www.reddit.com/r/cscareerquestions/comments/2povgz/what_is_the_best_programming_language_for_career/

Approaching the question

- Some web portals offer a summarized view for the query.
However, if we intend to create our own version, we need to perform the following steps
 - **Correct (or crawl)** several job search websites, e.g., Indeed and Glassdoor;
 - **Focus** on programming-related career positions;
 - **Compile a list** of all the thread-pairs consisting of required programming languages and associated salaries
 - **Summarize** the search results.

Next

The screenshot shows a Quora search results page. At the top, there is a navigation bar with the Quora logo, a home icon with a red notification badge (1), a list icon, a pen icon, a profile icon, a bell icon with a red notification badge (5), and a search bar. Below the navigation bar, there are several search filters: "Learning Python", "Python (programming language)", "Learning to Program", "Programming Languages", "+1", and a small edit icon. The main title of the post is "What database should I learn after learning Python?". Below the title, there are interaction metrics: "Answer" (blue icon), "Follow · 46" (radio icon), "Request" (person icon), and a comment section showing "1" comment and a downvote arrow. There is also a share icon and a more options icon.

Quora

1

Search Quora

Learning Python Python (programming language) Learning to Program

Programming Languages +1

What database should I learn after learning Python?

Answer Follow · 46 Request 1 ⚓ ⚔

Approaching the question

- Using similar approach to the earlier question, we may have to
 - **Crawl** several job search websites
 - **Compile a list** of all the thread-pairs consisting of required programming languages and associated salaries
 - **Summarize** the search results.

Approaching the question

- If we attempt to perform all of the tasks while querying for answers, we may encounter several challenges, including
 - Slower processing times;
 - Difficulty in modifying the question and receiving instant answers;
 - And other potential issues.

A simple walkthrough

The screenshot shows a Quora search results page. At the top, there is a navigation bar with the Quora logo, a home icon with a red notification badge (1), a grid icon, a pencil icon, a person icon, a bell icon with a red notification badge (5), and a search bar. Below the navigation bar, there are several search filters: 'Learning Python', 'Python (programming language)', 'Learning to Program', 'Programming Languages', '+1', and a pencil icon. The main title of the post is 'What database should I learn after learning Python?'. Below the title, there are options to 'Answer', 'Follow · 46', 'Request', and interaction icons (comment, downvote, upvote, share, more).

- Let's play with a data snapshot provided in Kaggle
<https://www.kaggle.com/jobspikr/software-developer-job-listings-usa>
- It contains 10,000 records of job posting entires from Indeed.com

A simple walkthrough

Data Explorer
41.51 MB

software_developer_united_states_1971_20...

< software_developer_united_sta... ↴ ⌂

Detail Compact Column 3 of 21 columns ▾

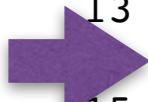
job_title	company...	job_descri...
Sr. Software Developer	Aerotek	The chosen Sr. Software Developer will be part of a larger engineering team developing software for ...
C# Lead Software Developer	3coast	Position: C# Lead Software Developer Location: Middletown, NJ Compensation: \$90,000 - \$110,0...
Senior Software Developer	s.com	Senior Software Developer Hoboken, N.J.

- Create a project and **store** the csv file in a folder named resource

Q: What DB should I learn after python?

- Sub question, e.g., How many positions required MySql and Python
- Create word list for each job entry

```
01 def get_and_clean_data():
02     data = pd.read_csv('resource/software_developer_united_states_1971_20191023_1.csv')
03     description = data['job_description']
04     cleaned_description = description.apply(lambda s: s.translate(str.maketrans('', '', string.punctuation + u'\xa0')))
05     cleaned_description = cleaned_description.apply(lambda s: s.lower())
06     cleaned_description = cleaned_description.apply(lambda s:
07         s.translate(str.maketrans(string.whitespace, ' '*len(string.whitespace), '')))
08     cleaned_description = cleaned_description.drop_duplicates()
09     return cleaned_description
10
11
12
13
14
15 def parse_job_description():
16     cleaned_description = get_and_clean_data()
17     cleaned_description = simple_tokenize(cleaned_description)
18     return cleaned_description
```



Q: What DB should I learn after python?

- Sub question: How many positions required MySql and Python

```
1 def count_python_mysql():
2     parsed_description = parse_job_description()
3     count_python = parsed_description.apply(lambda s: 'python' in s).sum()
4     count_mysql = parsed_description.apply(lambda s: 'mysql' in s).sum()
5     print('python: ' + str(count_python) + ' of ' + str(parsed_description.shape[0]))
6     print('mysql: ' + str(count_mysql) + ' of ' + str(parsed_description.shape[0]))
```

python: 1379 of 7583
mysql: 667 of 7583

Q: What db should I learn after python?

- What about other db

```
01 def parse_db():
02     html_doc = requests.get("https://db-engines.com/en/ranking").content
03     soup = BeautifulSoup(html_doc, 'html.parser')
04     db_table = soup.find("table", {"class": "dbi"})
05     all_db = [''.join(s.find('a').findAll(text=True, recursive=False)).strip() for s in
db_table.findAll("th", {"class": "pad-1"})]
06     all_db = list(dict.fromkeys(all_db))
07     db_list = all_db[:10]
08     db_list = [s.lower() for s in db_list]
09     db_list = [[x.strip() for x in s.split()] for s in db_list]
10     return db_list
```

```
>>>parse_db()
[['oracle'], ['mysql'], ['microsoft', 'sql', 'server'], ['postgresql'], ['mongodb'],
['redis'], ['elasticsearch'], ['ibm', 'db2'], ['sqlite'], ['microsoft', 'access']]
```

Q: Count of occurrences

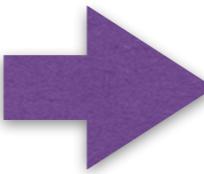
```
1 cleaned_db = parse_db()  
2 parsed_description = parse_job_description()  
3 raw = [None] * len(cleaned_db)  
4 for i,db in enumerate(cleaned_db):  
5     raw[i] = parsed_description.apply(lambda s: np.all([x in s for x in db])).sum()  
6     print(' '.join(db) + ': ' + str(raw[i]) + ' of ' + str(parsed_description.shape[0]))
```

```
oracle: 1392 of 7583  
mysql: 667 of 7583  
microsoft sql server: 868 of 7583  
postgresql: 261 of 7583  
mongodb: 296 of 7583  
redis: 106 of 7583  
elasticsearch: 161 of 7583  
ibm db2: 48 of 7583  
sqlite: 28 of 7583  
microsoft access: 256 of 7583
```

Q: What db should I learn after python?

```
1 with_python = [None] * len(cleaned_db)
2 for i,db in enumerate(cleaned_db):
3     with_python[i] = parsed_description.apply(lambda s: np.all([x in s for x in db]) and 'python' in s).sum()
4     print(' '.join(db) + ' + python: ' + str(with_python[i]) + ' of ' +
str(parsed_description.shape[0]))
```

oracle: 1392 of 7583
mysql: 667 of 7583
microsoft sql server: 868 of 7583
postgresql: 261 of 7583
mongodb: 296 of 7583
redis: 106 of 7583
elasticsearch: 161 of 7583
ibm db2: 48 of 7583
sqlite: 28 of 7583
microsoft access: 256 of 7583



oracle + python: 243 of 7583
mysql + python: 207 of 7583
microsoft sql server + python: 51 of 7583
postgresql + python: 90 of 7583
mongodb + python: 111 of 7583
redis + python: 38 of 7583
elasticsearch + python: 73 of 7583
ibm db2 + python: 12 of 7583
sqlite + python: 7 of 7583
microsoft access + python: 28 of 7583

Q: What db should I learn after python?

```
1 for i, db in enumerate(cleaned_db):
2     print(' '.join(db) + ' + python: ' + str(with_python[i]) + ' of ' + str(raw[i]) + ' (' +
str(np.around(with_python[i] / raw[i]*100,2)) + '%)')
```

oracle + python: 243 of 1392 (17.46%)
mysql + python: 207 of 667 (31.03%)
microsoft sql server + python: 51 of 868 (5.88%)
postgresql + python: 90 of 261 (34.48%)
mongodb + python: 111 of 296 (37.5%)
redis + python: 38 of 106 (35.85%)
elasticsearch + python: 73 of 161 (45.34%)
ibm db2 + python: 12 of 48 (25.0%)
sqlite + python: 7 of 28 (25.0%)
microsoft access + python: 28 of 256 (10.94%)

Q: What db should I learn after python?

- Some observations

oracle + python: 243 of 1392 (17.46%)

raw #1

mysql + python: 207 of 667 (31.03%)

raw #2

microsoft sql server + python: 51 of 868 (5.88%)

postgresql + python: 90 of 261 (34.48%)

mongodb + python: 111 of 296 (37.5%)

raw #3, % #2

redis + python: 38 of 106 (35.85%)

% #3

elasticsearch + python: 73 of 161 (45.34%)

% #1

ibm db2 + python: 12 of 48 (25.0%)

sqlite + python: 7 of 28 (25.0%)

microsoft access + python: 28 of 256 (10.94%)

To enable a quick search for other queries

- E.g.,
 - What DB should I learn after python?
 - What DB should I learn after java?
 - Which DB is in demand alongside oracle?
 - What programming language is in demand alongside python?
 - ...
- Indexing
 - Without indexing, it is impossible to process the queries for billion webpages x million users per seconds

Term-document incidence matrices

```
1 lang = [['java'],['python'],['c'],['kotlin'],['swift'],['rust'],['ruby'],['scala'],['julia'],
['lua']]
2 parsed_description = parse_job_description()
3 parsed_db = parse_db()
4 all_terms = lang + parsed_db
5 query_map = pd.DataFrame(parsed_description.apply(lambda s: [1 if np.all([d in s for d in db])
else 0 for db in all_terms]).values.tolist(), columns=[''.join(d) for d in all_terms])
```

	java	python	c	kotlin	...	sqlite	microsoft access
0	0	0	1	0	...	0	0
1	0	0	1	0	...	0	0
2	0	0	0	0	...	0	0
3	0	0	0	0	...	0	0
4	0	0	1	0	...	0	0
...
7578	1	1	0	0	...	0	0
7579	1	0	0	0	...	0	0
7580	0	0	1	0	...	0	0
7581	1	0	0	0	...	0	0
7582	1	0	0	0	...	0	0

- 1 = contain term, 0 not contain term.

Querying

- What DB should I learn after java?
- Which DB is in demand alongside oracle?
- What programming language is in demand alongside python?
 - Create a 0-1 vector using bitwise and/or
 - Start with something like:

```
query_map[query_map['java'] > 0].apply(lambda s: np.where(s==1)[0],  
axis=1).apply(lambda s: list(query_map.columns[s]))  
  
10      [java, python, c, oracle, mysql, mongodb]  
11                      [java]  
12                      [java, swift, redis]  
16                      [java, c, swift, oracle]  
19                      [java, python]  
...  
7576                      [java, oracle]  
7578      [java, python, oracle, mysql]  
7579                      [java, oracle]  
7581      [java, oracle, mysql]  
7582                      [java, mongodb]
```

Issue

- Scalability — if the #documents reaches 1 billion and each page contain 1000 words,
 - how large the 0/1 metrics would be?
 - How can we store them in the memory for quick searching?
- **Note:** ~ 4.22 billion is the #websites in the world in 2023
(<https://www.worldwidewebsize.com/>)

Some thoughts

- We need a better tokenizer.
- The 0/1 matrix is extremely sparse -> We can better represent it.

Inverted index

- The same as:

Index

Armada Portrait at Greenwich ➤ 7, 44–5, 57–8, 65, 74–7
Armada Portrait at the National Portrait Gallery ➤ 9, 45, 65, 67, 69
Armada Portrait at Woburn ➤ 8, 45, 65

Battle of Gravelines ➤ 25, 30, 45, 46
Boleyn, Anne ➤ 10

Calais, pursuit to ➤ 24, 25, 30
Catherine of Aragon ➤ 10
Charles Howard, 1536–1624, 1st Earl of Nottingham (David Mytens the Elder) ➤ 26

Darnley pattern ➤ 48, 58, 65, 66
de Heere, Lucas ➤ 17

de la Cruz, Juan Pantoja ➤ 38
Denizot, Nicholas ➤ 12
Ditchley portrait ➤ 60, 67, 68, 69
Drake, Francis ➤ 23, 26, 36, 47–8, 51–2, 53

Edward VI ➤ 11, 13–14
Edward VI (1537–1553) (after Hans Holbein) ➤ 15
Elizabeth (Crispin van de Passe the Elder) ➤ 64
Elizabeth I (Nicholas Hilliard) ➤ 50
Elizabeth I (1533–1603) (English school) ➤ 49
Elizabeth I (1533–1603) (William Rogers) ➤ 40

Elizabeth I, the Ditchley portrait (Marcus Gheeraerts the Younger) ➤ 68
Elizabeth I, the Ermine portrait ➤ 59
Elizabeth I, the Rainbow portrait ➤ 70
Elizabeth I, the Siena Sieve portrait (Quentin Massys the Younger) ➤ 62
Elizabeth I, the Wanstead (or Peace) portrait (Marcus Gheeraerts the Younger) ➤ 57
English Ships and the Spanish Armada, August 1588 ➤ 46
Ermine portrait ➤ 58, 59, 60, 61
Expeditionis Hispanorum in Angliam ➤ 24, 30

Gheeraerts the Elder, Marcus ➤ 57, 58
Gheeraerts the Younger, Marcus ➤ 53, 68
Gower, George ➤ 8, 45, 47
Grey, Lady Jane ➤ 14–15

Henry VIII ➤ 10–13
Henry VIII (1491–1547) (studio of Hans Holbein) ➤ 11
Hilliard, Nicholas ➤ 47, 50, 51, 52, 58, 65
Holbein, Hans ➤ 11, 15, 66
Howard, Charles ➤ 24, 25, 26

James VI (James I of England) ➤ 23, 36–7, 38

launch of the fireships ➤ 28, 44
Launch of the fireships against the Spanish Armada, 7 August 1588 ➤ 28

Mary I (Mary Tudor) ➤ 10, 12, 14–18

Rainbow portrait ➤ 67, 69, 70
Raleigh, Walter ➤ 23
Rogers, William ➤ 40
route of the Armada around Britain and Ireland ➤ 30

Seymour, Jane ➤ 10–11
Seymour, Thomas ➤ 12, 13
Siena Sieve portrait ➤ 61, 62, 65, 66
Sir Francis Drake, 1540–96 (Marcus Gheeraerts the Younger) ➤ 53
Somerset House conference ➤ 38
Somerset House conference, 19 August 1604 (after Juan Pantoja de la Cruz) ➤ 38
speech at Tilbury ➤ 27, 66–7

Thomas Seymour, 1st Baron Seymour of Sudeley, c. 1509–49 ➤ 12
Tilbury speech ➤ 27, 66–7

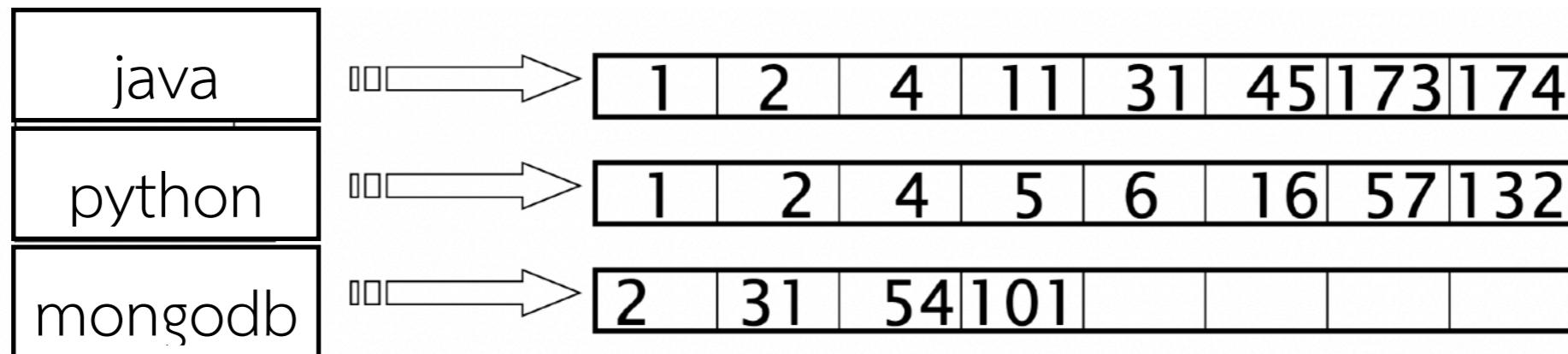
van de Passe the Elder, Crispin ➤ 64
Wanstead portrait ➤ 57, 58, 60, 65

Ref: <https://www.userdesignillustrationandtypesetting.com/book-design/index.html>

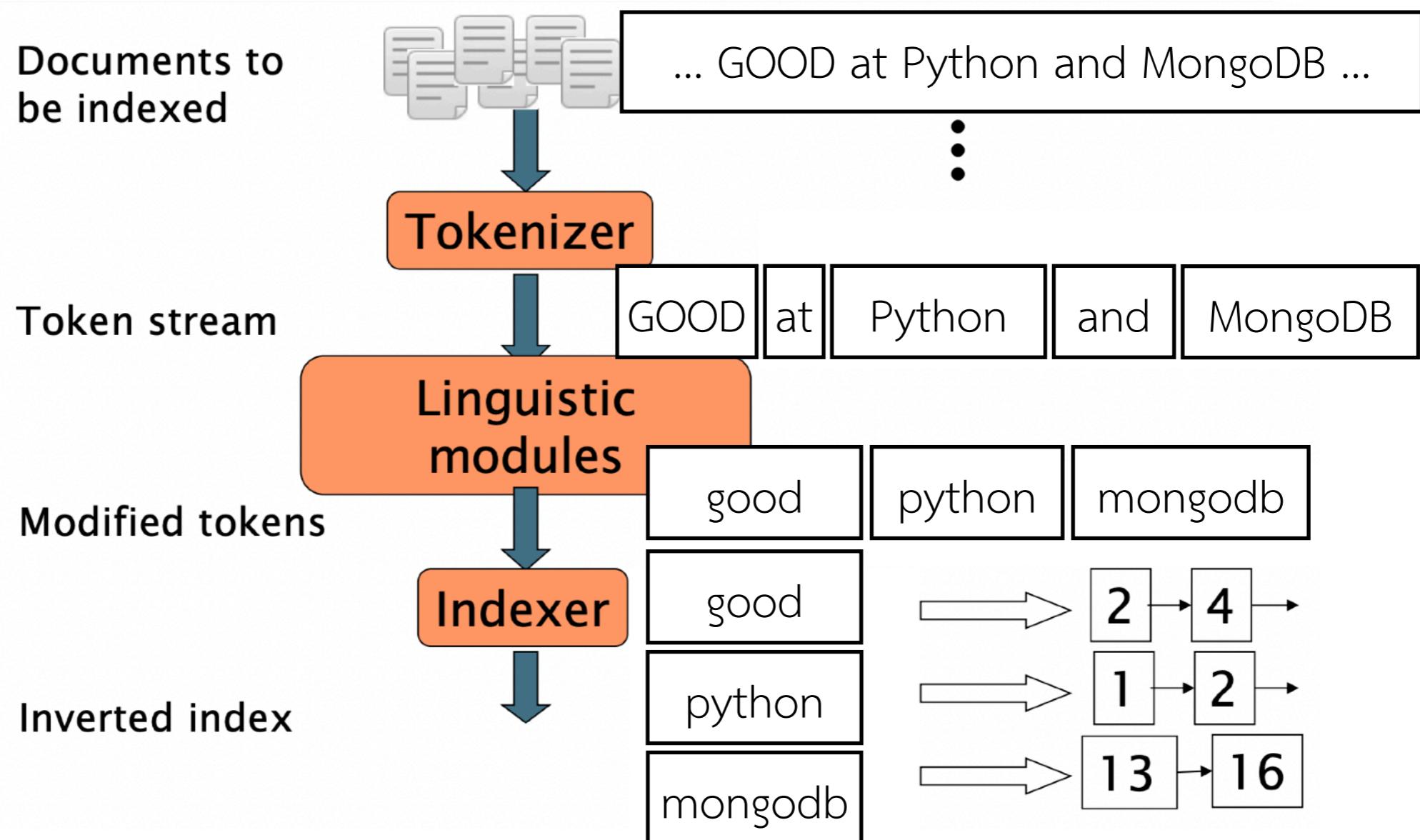
Inverted index

- For each term **t**, we store a list of all documents that contain **t**.
 - Identify each document by a **docID**, i.e., a document serial number.

- E.g.,

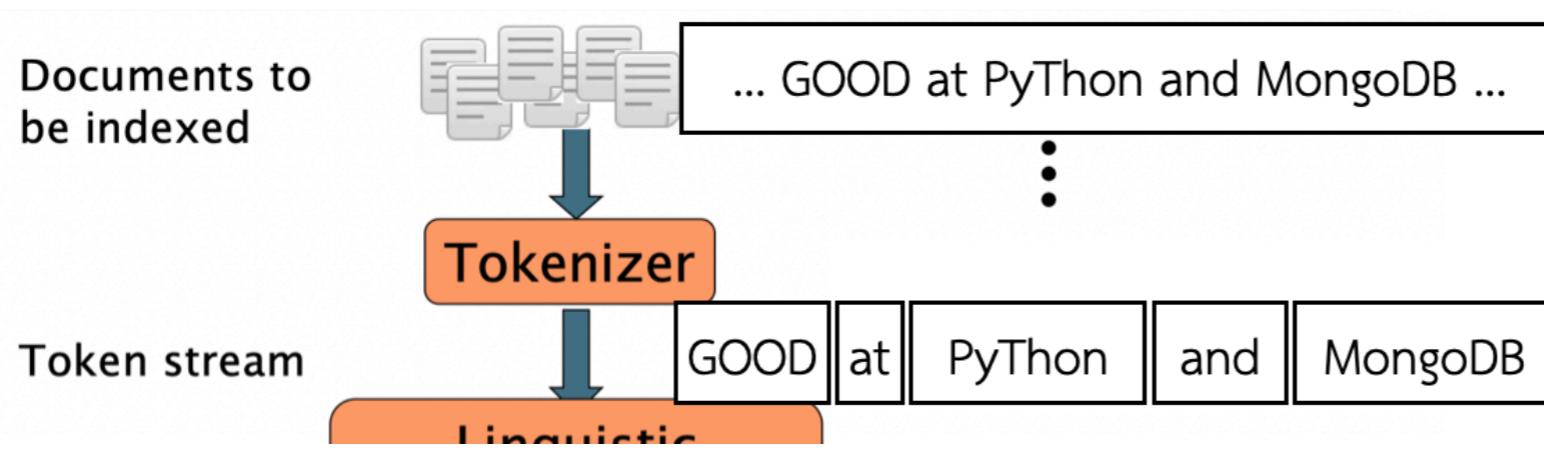


Inverted index construction



Each component

- Tokenization
 - Cut character sequence into word tokens



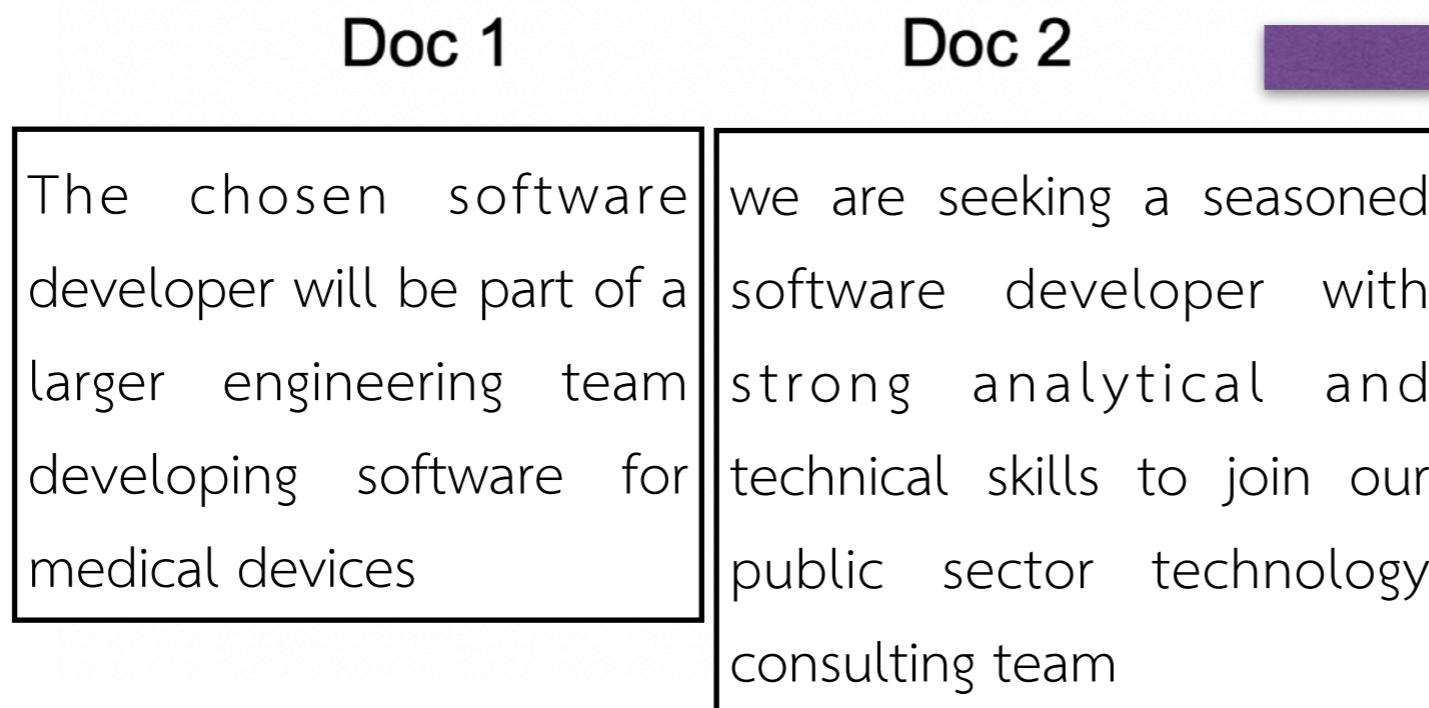
- Normalization
 - Map text and query term to same form,
e.g., S.Q.L. and sql must refer to the same thing.

Each component

- Stemming | lemmatizing
 - We may wish different forms of a root to match,
e.g., **developed** and **developing** must refer to the same thing.
- Stop words
 - We may omit very common words (or not)
e.g., **and, or, not, a, an ,the**

Each component

- Indexer steps: Token sequence
 - Sequence of (Modified token, Document ID) pairs.



Each component

```
01 str1 = 'the chosen software developer will be part of a larger engineering team developing  
software for medical devices.'  
02 str2 = 'we are seeking a seasoned software developer with strong analytical and technical skills  
to join our public sector technology consulting team.'  
03  
04 import nltk  
05 nltk.download('stopwords')  
06 nltk.download('punkt')  
07 from nltk.corpus import stopwords  
08 from nltk.tokenize import word_tokenize  
09 from nltk.stem import PorterStemmer  
10  
11 tokened_str1 = word_tokenize(str1)  
12 tokened_str2 = word_tokenize(str2)  
13  
14 tokened_str1 = [w for w in tokened_str1 if len(w) > 2]  
15 tokened_str2 = [w for w in tokened_str2 if len(w) > 2]  
16  
17 no_sw_str1 = [word for word in tokened_str1 if not word in stopwords.words()]  
18 no_sw_str2 = [word for word in tokened_str2 if not word in stopwords.words()]  
19  
20 ps = PorterStemmer()  
21 stemmed_str1 = np.unique([ps.stem(w) for w in no_sw_str1])  
22 stemmed_str2 = np.unique([ps.stem(w) for w in no_sw_str2])  
23  
24 full_list = np.sort(np.concatenate([stemmed_str1, stemmed_str2]))
```

Several components of the nltk library

Tokenize

Remove low frequency

Remove Stop words

Stemming

full_list

```
['analyt', 'chosen', 'consult', 'develop', 'develop', 'devic', 'engin',  
'join', 'larger', 'medic', 'part', 'public', 'season','sector', 'seek', 'skill',  
'softwar', 'softwar', 'strong', 'team','team', 'technic', 'technolog']
```

Each component

- Indexer steps: Sort
 - Sort by terms
 - At least conceptually
 - And then docID



Term	DocID
chosen	1
softwar	1
part	1
larger	1
engin	1
team	1
develop	1
medic	1
devic	1
seek	2
season	2
softwar	2
develop	2
strong	2
analyt	2
technic	2
skill	2
join	2
public	2
sector	2
seek	2
skill	2
softwar	2
softwar	1
strong	2
team	2
team	1
technic	2
technolog	2

A large purple arrow points from the first table to the second.

Term	DocID
analyt	2
chosen	1
consult	2
develop	2
develop	1
devic	1
engin	1
join	2
larger	1
medic	1
part	1
public	2
season	2
sector	2
seek	2
skill	2
softwar	2
softwar	1
strong	2
team	2
team	1
technic	2
technolog	2

Each component

- Indexer steps: Dictionary & Postings

- Multiple term entries in a single document are merged.
- Split into Dictionary and Postings
- Doc. frequency information is added.

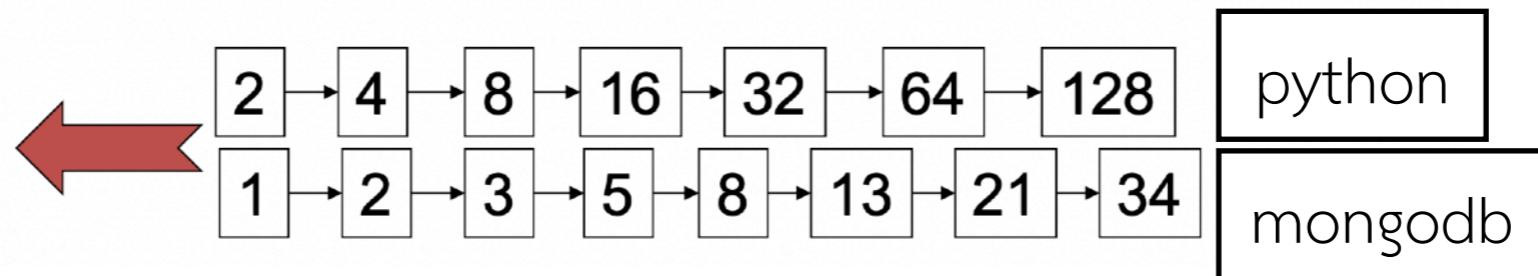
Why frequency?
Will discuss later.

Term	DocID
analyt	2
chosen	1
consult	2
develop	2
develop	1
devic	1
engin	1
join	2
larger	1
medic	1
part	1
public	2
season	2
sector	2
seek	2
skill	2
softwar	2
softwar	1
strong	2
team	2
team	1
technic	2
technolog	2

Term	doc count	posting lists
analyt	1	→ 1
chosen	1	→ 1
consult	1	→ 1
develop	2	→ 1 → 2
devic	1	→ 1
engin	1	→ 1
join	2	→ 1 → 2
larger	1	→ 1
medic	1	→ 1
part	1	→ 1
public	1	→ 2
season	1	→ 2
sector	1	→ 2
seek	1	→ 2
skill	1	→ 2
softwar	2	→ 1 → 2
strong	1	→ 2
team	2	→ 1 → 2
technic	1	→ 2
technolog	1	→ 2

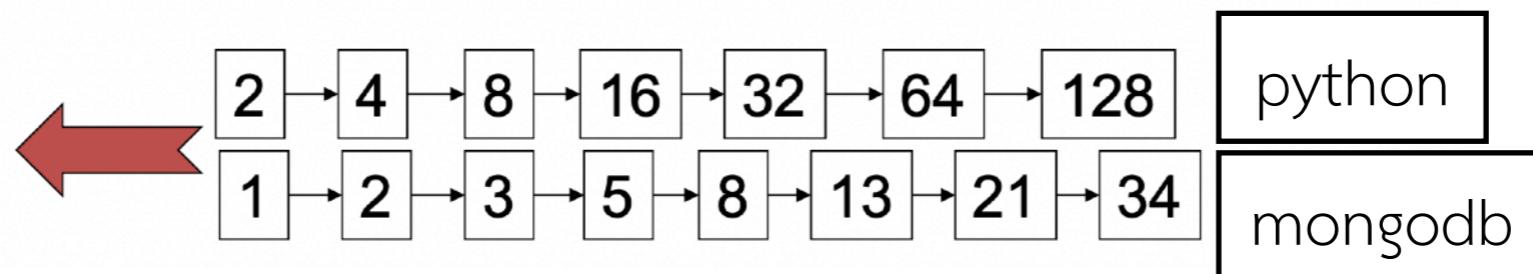
Querying

- AND
 - E.g., Python and MongoDB
 - Locate **python** in the **Dictionary** and get its **Posting list**;
 - Locate **mongodb** in the **Dictionary** and get its **Posting list**;
 - Merge the two **Posting lists**, i.e., set intersection.



Querying

- OR
 - E.g., Python and MongoDB
 - Locate **python** in the **Dictionary** and get its **Posting list**;
 - Locate **mongodb** in the **Dictionary** and get its **Posting list**;
 - Merge the two **Posting lists**, i.e., set union.



Query optimization example

- Process in order of increasing frequencies:
 - Start with smallest set, then keep cutting further.
 - E.g., (python OR ruby) AND (mysql or sqlite)

java: 3268 of 7583

python: 1379 of 7583

c: 3434 of 7583

kotlin: 45 of 7583

swift: 124 of 7583

rust: 10 of 7583

ruby: 346 of 7583

scala: 108 of 7583

julia: 2 of 7583

lua: 20 of 7583

oracle: 1392 of 7583

mysql: **667** of 7583

microsoft sql server: 868 of 7583

postgresql: 261 of 7583

mongodb: 296 of 7583

db2: 130 of 7583

redis: 106 of 7583

elasticsearch: 161 of 7583

sqlite: **28** of 7583

microsoft access: 256 of 7583

More general optimization

- Get document frequency's for all terms.
- Estimate the size of each OR by the sum of its document frequency's (conservative).
- Process in increasing order of OR sizes.

Assignment

- Show that with the approach demonstrated in the class you are able to answer the following questions:
 1. What DB should I learn after java? (10%)
 2. Which DB is in demand alongside oracle? (10%)
 3. Which programming language is in demand alongside python? (10%)
- Create one question beginning with **WHY** that can be answered using the data we've analyzed during this class. Then, provide your solution to the question. (70%)

Time for questions