

Inverted Indexing for Text Retrieval

Three components of the web search problem

- Gathering web content
 - ❖ Web crawling
 - Construction of the inverted index
 - ❖ Indexing
 - Ranking documents given a query
 - ❖ Retrieval
-
- First two steps are typically carried out off-line
 - The retrieval step needs to be operated in real time

What is inverted index?

- First, what is index?

Index

A

addictions 35, 39
the Arctic 68, 69

B

the bar 48, 68, 69, 75, 81, 89, 90, 132, 149
bush tips 36, 38, 52, 57, 58, 65, 66, 79, 129
business 34, 39, 48, 76, 77, 116, 128, 132, 145, 156

C

city life 41, 54, 72, 74, 98, 108, 116, 135, 146, 147

D

dog's life 12, 16, 21, 22, 24, 25, 32, 40, 43, 53, 57, 61, 62, 66, 71, 75, 86, 87, 89, 91, 92, 95, 96, 98, 100, 101, 102, 103, 112, 115, 125, 126, 129, 144, 152, 160

E

the environment 26, 33, 62, 77, 78, 82, 85, 121, 124, 128, 134, 136, 137, 138, 139, 147, 148, 149, 150, 151, 152, 160, 163
explorers 17, 28, 29, 77, 114

F

fall 13, 83
fishing 6, 9, 10, 19, 46, 78, 127, 157, 159

flying 22, 23

G

government 12, 17, 25, 43, 49, 57, 63, 68, 70, 76, 79, 80, 99, 114, 126, 127, 133, 140, 143, 146, 151, 153, 155

grouse 24, 27, 163

guiding 17, 61, 86, 122, 128, 136, 158

H

health 30
hippies 15, 35, 36, 37, 88
horses 47, 122, 144

hunting 14, 31, 32, 46, 61, 67, 70, 72, 85, 86, 106, 107, 124, 127, 141, 151, 161, 162

K

kids 24, 30, 58, 59, 74, 123, 137

M

marriage 23, 33, 36, 49, 52, 56, 60, 66, 68, 71, 75, 76, 89, 90, 96, 97, 100, 102, 124, 141, 154, 157
mechanics 18, 66, 87, 145
money 32

mushing 1, 2, 3, 4, 5, 9, 10, 18, 20, 21, 34, 37, 43, 53, 55, 73, 74, 75, 89, 94, 98, 109, 112, 150

O

oldtimers 43, 93, 107, 126, 156, 158

P

philosophy 94, 102, 108, 109, 113, 115, 116, 121, 123, 130, 132, 134, 142, 144, 146, 152, 154, 155, 156, 157, 159, 161, 162, 164

police 11, 13, 27, 34, 44

politics 6, 44, 56, 67, 71, 73, 125, 139, 140, 142

prospecting 18, 29, 31, 64, 104, 123, 130

punks 16, 19, 47, 108, 130

R

ravens 21, 30, 47, 67, 71, 95, 104, 124, 134, 137, 138, 139, 140, 165

religion 7, 38, 45, 58

romance 4, 5, 16, 23, 35, 42, 50, 52, 55, 56, 71, 114, 145

S

scientists 11, 25, 28, 38, 63, 65, 67, 90, 120, 122, 127, 135, 137, 138, 139, 140, 150, 155, 163, 164
snowmachines 20, 149

southerners 49, 60, 62, 63, 72, 74, 76, 80, 81, 82, 99, 101, 103, 113, 120, 122, 131, 133, 136, 142, 146, 152, 153, 154, 158, 159, 160
summer 10, 27, 45, 64, 157

T

tourism 8, 21, 26, 33, 41, 44, 45, 46, 64, 80, 81, 83, 120, 125, 133, 135, 143, 158

tradition 84, 98

trapping 14, 19, 32, 56, 60, 72, 92, 94, 97, 107, 109, 112, 113, 128, 153, 165

W

weather 20, 50, 93, 101, 115

winter 50, 51, 107

Example of inverted index

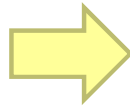
Doc 1
one fish, two fish

Doc 2
red fish, blue fish

Doc 3
cat in the hat

Doc 4
green eggs and ham

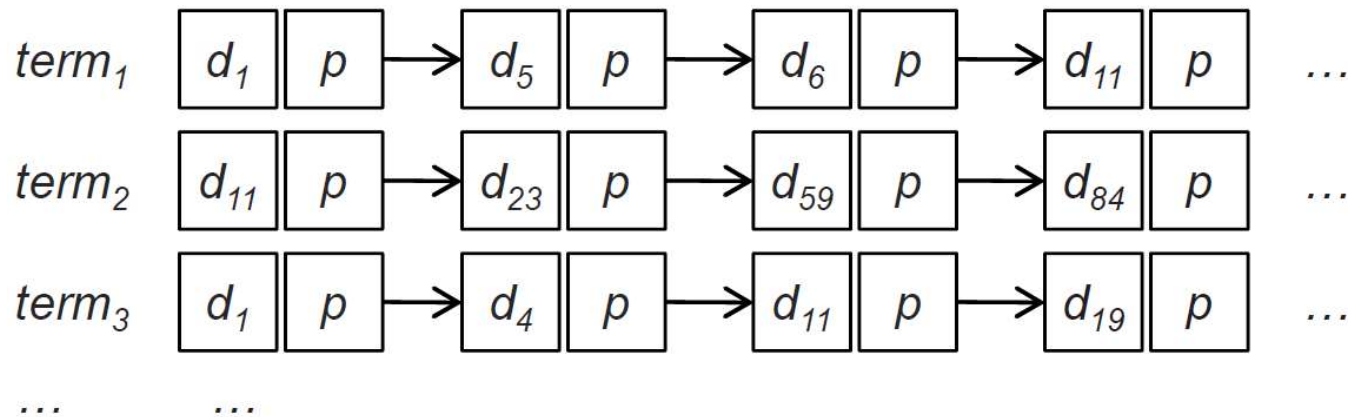
	1	2	3	4
blue		1		
cat			1	
egg				1
fish	1	1		
green				1
ham				1
hat			1	
one	1			
red		1		
two	1			



blue	→	2
cat	→	3
egg	→	4
fish	→	1 → 2
green	→	4
ham	→	4
hat	→	3
one	→	1
red	→	2
two	→	1

More abstract view of inverted index

terms **postings**



- An inverted index consists of posting lists
- A posting list is comprised of individual postings
 - ❖ Each posting consists of a document id and a payload
 - Payload example: the occurrence frequency of the term in the corresponding document
 - ❖ Generally, postings are sorted by document id

Baseline implementation of inverted indexing

```
1: class MAPPER
2:   procedure MAP(docid  $n$ , doc  $d$ )
3:      $H \leftarrow$  new ASSOCIATIVEARRAY
4:     for all term  $t \in$  doc  $d$  do
5:        $H\{t\} \leftarrow H\{t\} + 1$ 
6:     for all term  $t \in H$  do
7:       EMIT(term  $t$ , posting  $\langle n, H\{t\} \rangle$ )

1: class REDUCER
2:   procedure REDUCE(term  $t$ , postings [ $\langle n_1, f_1 \rangle, \langle n_2, f_2 \rangle \dots$ ])
3:      $P \leftarrow$  new LIST
4:     for all posting  $\langle a, f \rangle \in$  postings [ $\langle n_1, f_1 \rangle, \langle n_2, f_2 \rangle \dots$ ] do
5:        $P.ADD(\langle a, f \rangle)$ 
6:      $P.SORT()$ 
7:     EMIT(term  $t$ , postings  $P$ )
```

Illustration of the baseline algorithm

Map

Doc 1
one fish, two fish

one

1	1
---	---

two

1	1
---	---

fish

1	2
---	---

Doc 2
red fish, blue fish

red

2	1
---	---

blue

2	1
---	---

fish

2	2
---	---

Doc 3
cat in the hat

cat

3	1
---	---

hat

3	1
---	---

Shuffle and Sort: aggregate values by keys

Reduce

cat

3	1
---	---

fish

1	2
---	---

2	2
---	---

one

1	1
---	---

red

2	1
---	---

blue

2	1
---	---

hat

3	1
---	---

two

1	1
---	---

Inverted Indexing: Pseudo-Code

```
1: class MAPPER
2:   procedure MAP(docid  $n$ , doc  $d$ )
3:      $H \leftarrow$  new ASSOCIATIVEARRAY
4:     for all term  $t \in$  doc  $d$  do
5:        $H\{t\} \leftarrow H\{t\} + 1$ 
6:     for all term  $t \in H$  do
7:       EMIT(term  $t$ , posting  $\langle n, H\{t\} \rangle$ )

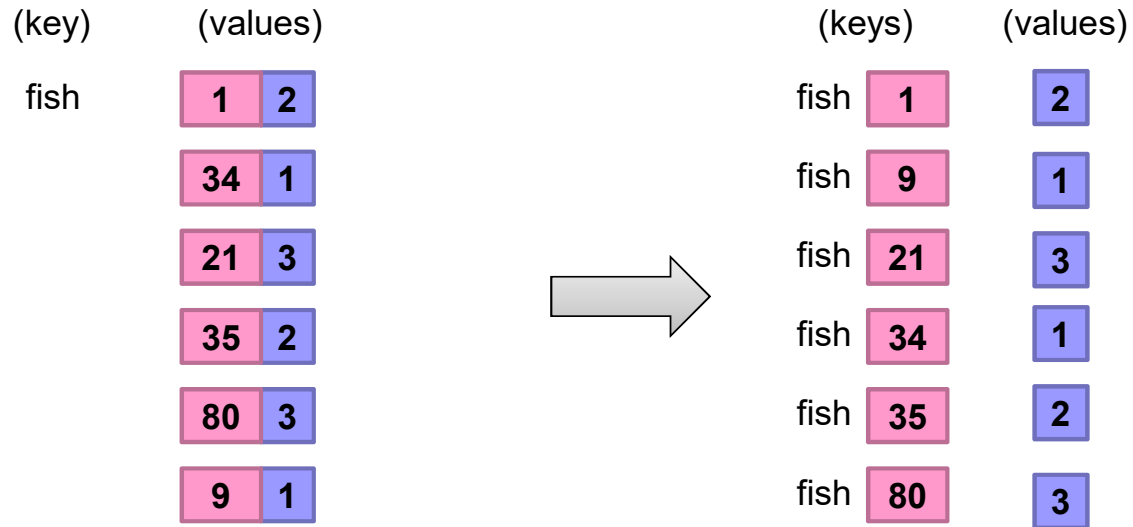
1: class REDUCER
2:   procedure REDUCE(term  $t$ , postings  $[\langle a_1, f_1 \rangle, \langle a_2, f_2 \rangle \dots]$ )
3:      $P \leftarrow$  new LIST
4:     for all posting  $\langle a, f \rangle \in$  postings  $[\langle a_1, f_1 \rangle, \langle a_2, f_2 \rangle \dots]$  do
5:       APPEND( $P, \langle a, f \rangle$ )
6:       SORT( $P$ )
7:       EMIT(term  $t$ , postings  $P$ )
```

What's the problem?

Scalability issue of the baseline implementation

- Initial implementation: terms as keys, postings as values
 - ❖ Reducers must buffer all postings associated with key (to sort)
 - ❖ What if we run out of memory to buffer postings?

Another try



- Value-to-key conversion

Revised implementation

```
1: class MAPPER
2:   method MAP(docid  $n$ , doc  $d$ )
3:      $H \leftarrow \text{new ASSOCIATIVEARRAY}$ 
4:     for all term  $t \in \text{doc } d$  do
5:        $H\{t\} \leftarrow H\{t\} + 1$ 
6:     for all term  $t \in H$  do
7:        $\text{EMIT}(\text{tuple } \langle t, n \rangle, \text{tf } H\{t\})$ 
```

```
1: class REDUCER
2:   method INITIALIZE
3:      $t_{prev} \leftarrow \emptyset$ 
4:      $P \leftarrow \text{new POSTINGSLIST}$ 
5:   method REDUCE(tuple  $\langle t, n \rangle$ ,  $\text{tf}[f]$ )
6:     if  $t \neq t_{prev} \wedge t_{prev} \neq \emptyset$  then
7:        $\text{EMIT}(\text{term } t, \text{postings } P)$ 
8:        $P.\text{RESET}()$ 
9:        $P.\text{ADD}(\langle n, f \rangle)$ 
10:     $t_{prev} \leftarrow t$ 
11:   method CLOSE
12:      $\text{EMIT}(\text{term } t, \text{postings } P)$ 
```