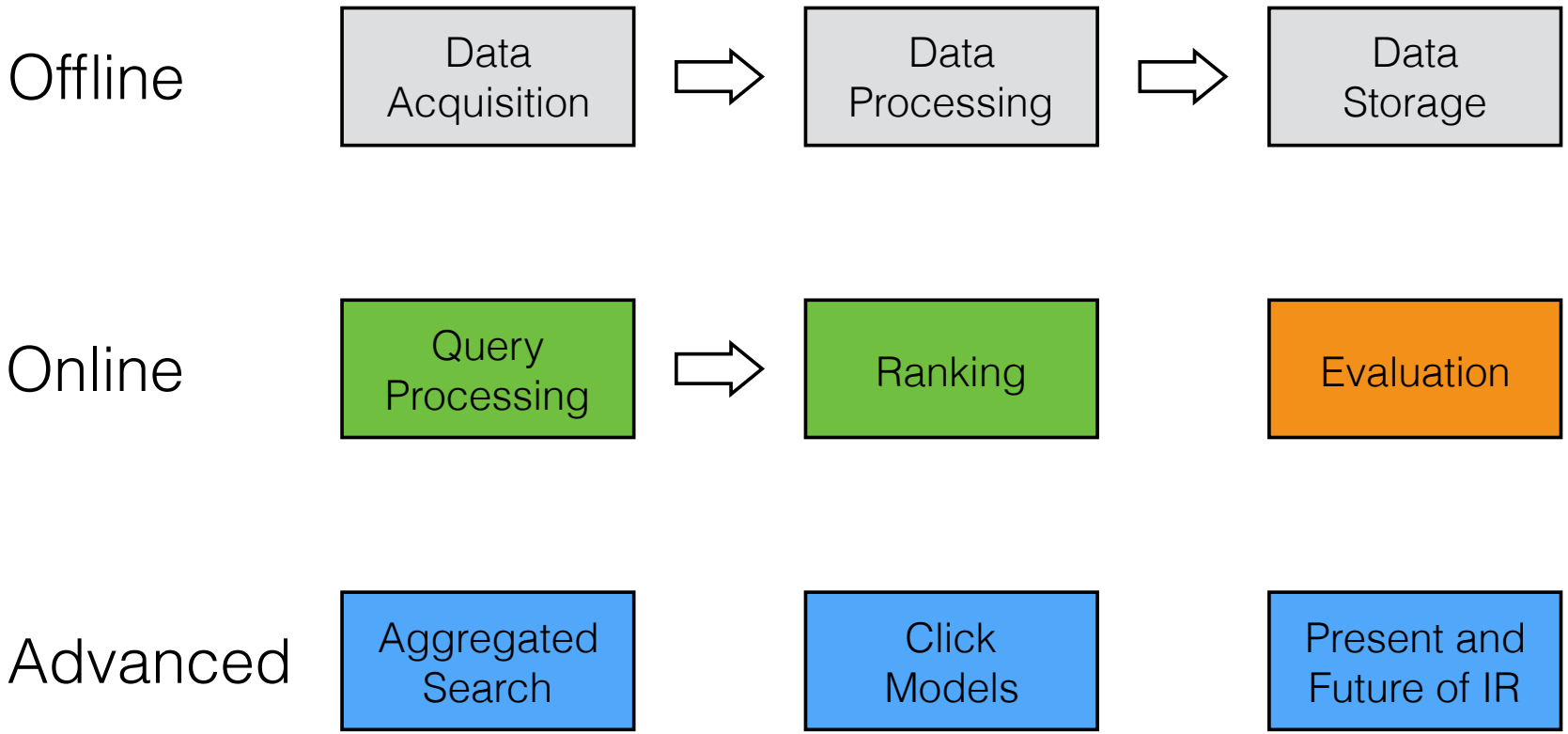
Information Retrieval

Data Processing and Storage

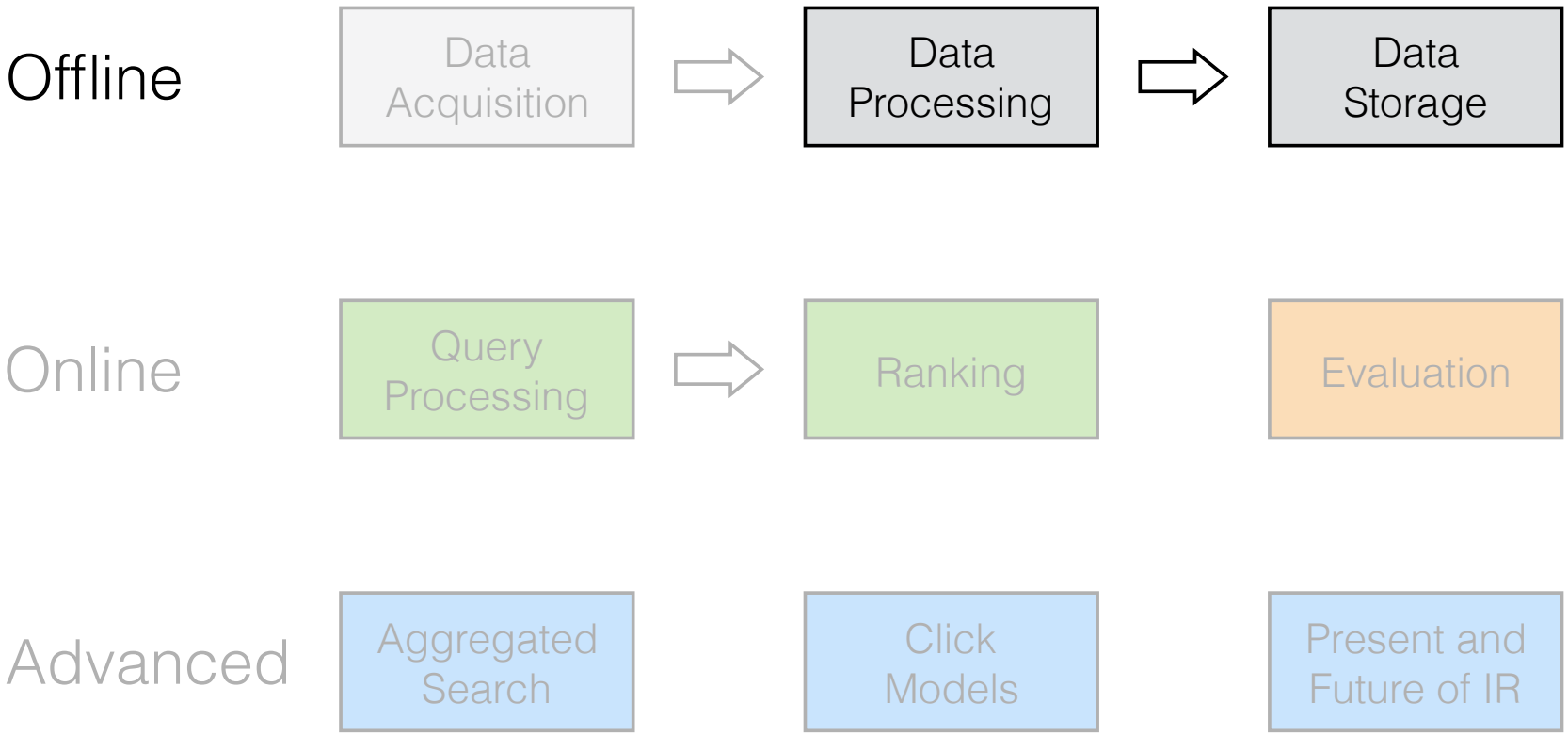
Ilya Markov
i.markov@uva.nl

University of Amsterdam

Course overview



This lecture



Outline

Outline

Outline

Data processing pipeline

$$\begin{aligned} \text{Text document} &\Longrightarrow \text{Lexical analysis} \\ &\Longrightarrow \text{Stop-word removal} \\ &\Longrightarrow \text{Stemming} \end{aligned}$$

Example

1. To prepare a text for indexing, one needs to split it into tokens, remove stop-words and perform stemming.
2. To prepare a text for indexing one needs to split it into tokens remove stop words and perform stemming
3. prepare indexing needs split tokens remove stop perform stemming
4. prepar index need split token remov stop perform stem

Lexical analysis

1. Remove punctuation
2. Decide on what a "word" is
3. Lowercase everything

Stop-word removal

- Dictionary-based
  - Create a dictionary of stop-words
  - Remove words that occur in this dictionary
- Frequency-based
  - Set a frequency threshold f
  - Remove words with the frequency higher than f

Outline

Stemming

1. Algorithmic
2. Dictionary-based
3. Hybrid

Algorithmic stemming (Porter stemmer)

Step 1a:

- Replace ***sses*** by ***ss*** (e.g., stresses → stress).
- Delete ***s*** if the preceding word part contains a vowel not immediately before the ***s*** (e.g., gaps → gap but gas → gas).
- Replace ***ied*** or ***ies*** by ***i*** if preceded by more than one letter, otherwise by ***ie*** (e.g., ties → tie, cries → cri).
- If suffix is ***us*** or ***ss*** do nothing (e.g., stress → stress).

Step 1b:

- Replace ***eed***, ***eedly*** by ***ee*** if it is in the part of the word after the first non-vowel following a vowel (e.g., agreed → agree, feed → feed).
- Delete ***ed***, ***edly***, ***ing***, ***ingly*** if the preceding word part contains a vowel, and then if the word ends in ***at***, ***bl***, or ***iz*** add ***e*** (e.g., fished → fish, pirating → pirate), or if the word ends with a double letter that is not ***ll***, ***ss*** or ***zz***, remove the last letter (e.g., falling→ fall, dripping → drip), or if the word is short, add ***e*** (e.g., hoping → hope).
- Whew!

Croft et al., "Search Engines, Information Retrieval in Practice"

Dictionary-based stemming

- Store lists of related words in a dictionary
- Can recognize the relation between “is”, “be”, “was”
- New-words problem

Hybrid stemming (Krovetz stemmer)

- Approach
  1. Check the word in a dictionary
  2. If present, either leave it as is or replace with exception
  3. If not present, check for suffixes that could be removed
  4. After removal, check the dictionary again
- Produces words not stems
- Comparable effectiveness with the Porter stemmer

Stemming example

Original text:
Document will describe marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales.

Porter stemmer:
document describ market strategi carri compani agricultur chemic report predict market share chemic report market statist agrochem pesticid herbicid fungicid insecticid fertil predict sale market share stimul demand price cut volum sale

Krovetz stemmer:
document describe marketing strategy carry company agriculture chemical report prediction market share chemical report market statistic agrochemic pesticide herbicide fungicide insecticide fertilizer predict sale stimulate demand price cut volume sale

Croft et al., "Search Engines, Information Retrieval in Practice"

Outline

Example

To be or not to be...

Dealing with phrases

1. Detect **noun phrases** using a part-of-speech tagger
   - sequences of nouns
   - adjectives followed by nouns
2. Detect phrases at the query processing time
   - Use an index with word positions
   - Will be discussed next
3. Use frequent **n-grams**, e.g., bigrams and trigrams

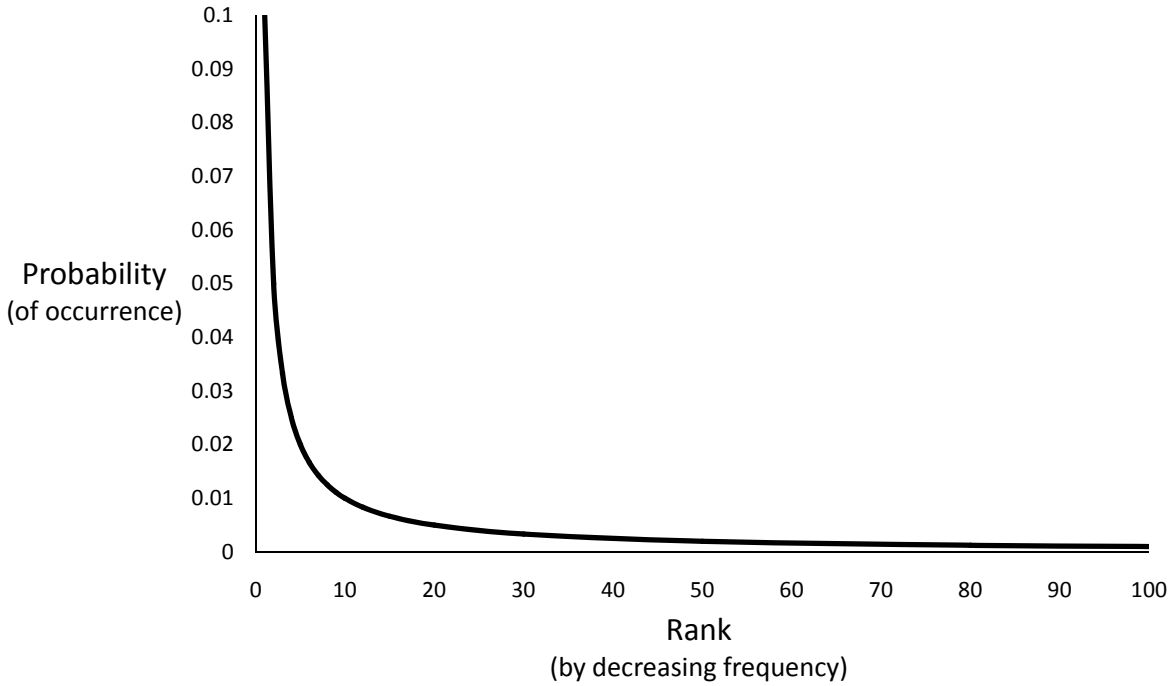
Example noun phrases

TREC data		Patent data	
Frequency	*Phrase*	*Frequency*	*Phrase*
65824	united states	975362	present invention
61327	article type	191625	u.s. pat
33864	los angeles	147352	preferred embodiment
18062	hong kong	95097	carbon atoms
17788	north korea	87903	group consisting
17308	new york	81809	room temperature
15513	san diego	78458	seq id
15009	orange county	75850	brief description
12869	prime minister	66407	prior art
12799	first time	59828	perspective view
12067	soviet union	58724	first embodiment
10811	russian federation	56715	reaction mixture
9912	united nations	54619	detailed description
8127	southern california	54117	ethyl acetate
7640	south korea	52195	example 1
7620	end recording	52003	block diagram
7524	european union	46299	second embodiment
7436	south africa	41694	accompanying drawings
7362	san francisco	40554	output signal
7086	news conference	37911	first end
6792	city council	35827	second end
6348	middle east	34881	appended claims
6157	peace process	33947	distal end
5955	human rights	32338	cross-sectional view
5837	white house	30193	outer surface

Croft et al., "Search Engines, Information Retrieval in Practice"

Outline

Zipf's law

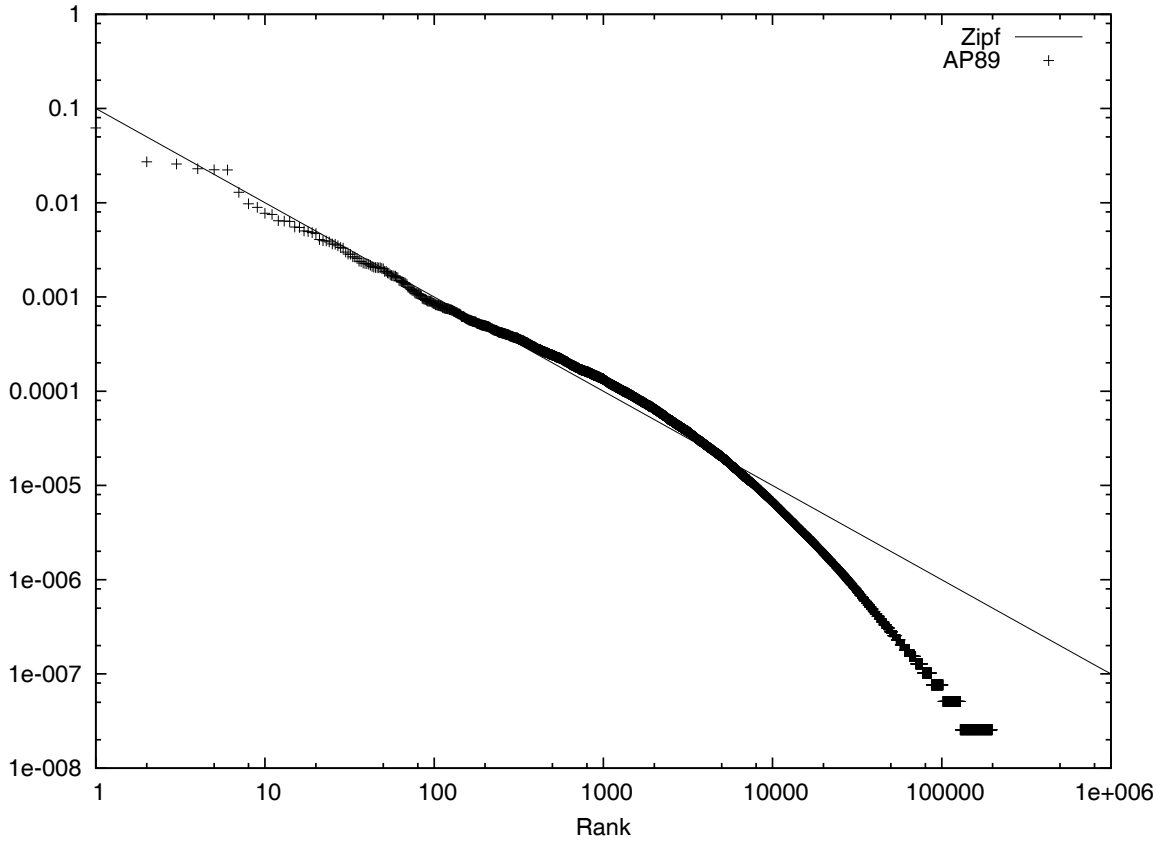


$$rank \cdot freq = const$$

$$rank \cdot P_r = const'$$

Croft et al., "Search Engines, Information Retrieval in Practice"

Zipf's law vs. real data



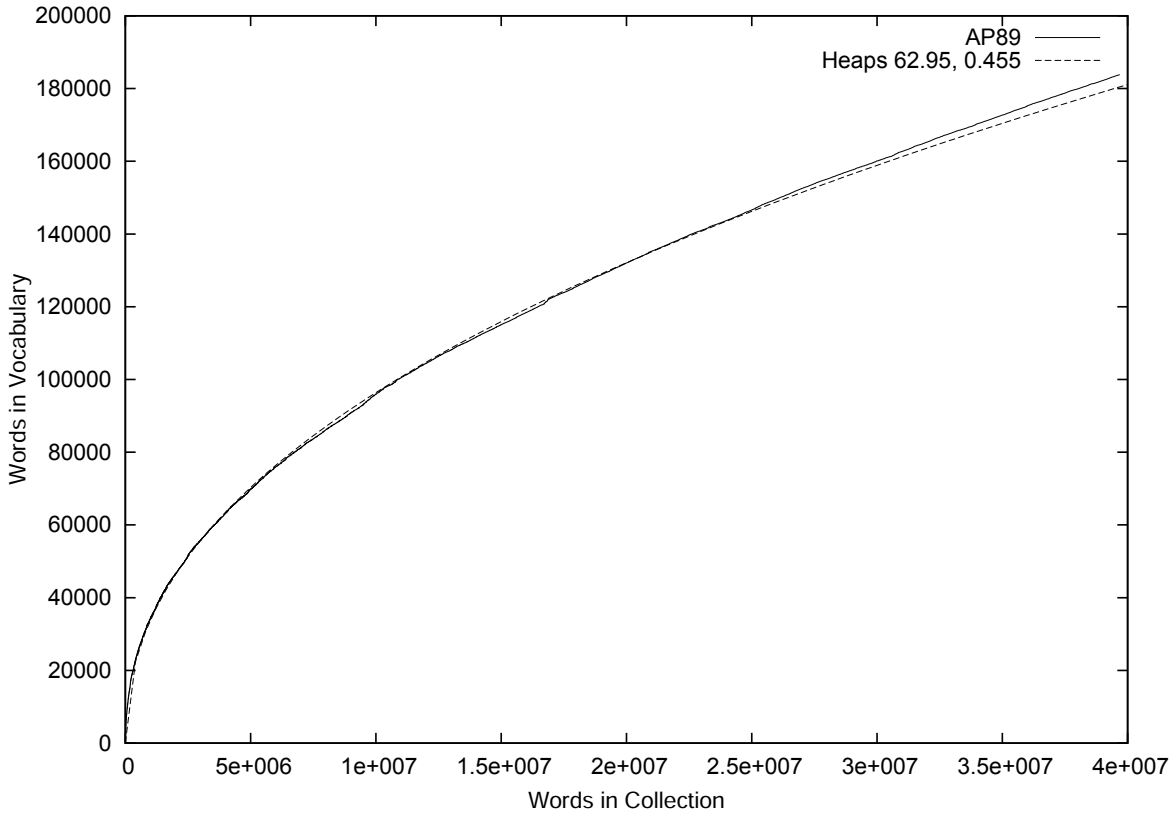
Croft et al., "Search Engines, Information Retrieval in Practice"

Zipf's law example

Word	*Freq.*	r	$P_r(\%)$	$r.P_r$	*Word*	*Freq*	r	$P_r(\%)$	$r.P_r$
the	2,420,778	1	6.49	0.065	has	136,007	26	0.37	0.095
of	1,045,733	2	2.80	0.056	are	130,322	27	0.35	0.094
to	968,882	3	2.60	0.078	not	127,493	28	0.34	0.096
a	892,429	4	2.39	0.096	who	116,364	29	0.31	0.090
and	865,644	5	2.32	0.120	they	111,024	30	0.30	0.089
in	847,825	6	2.27	0.140	its	111,021	31	0.30	0.092
said	504,593	7	1.35	0.095	had	103,943	32	0.28	0.089
for	363,865	8	0.98	0.078	will	102,949	33	0.28	0.091
that	347,072	9	0.93	0.084	would	99,503	34	0.27	0.091
was	293,027	10	0.79	0.079	about	92,983	35	0.25	0.087
on	291,947	11	0.78	0.086	i	92,005	36	0.25	0.089
he	250,919	12	0.67	0.081	been	88,786	37	0.24	0.088
is	245,843	13	0.65	0.086	this	87,286	38	0.23	0.089
with	223,846	14	0.60	0.084	their	84,638	39	0.23	0.089
at	210,064	15	0.56	0.085	new	83,449	40	0.22	0.090
by	209,586	16	0.56	0.090	or	81,796	41	0.22	0.090
it	195,621	17	0.52	0.089	which	80,385	42	0.22	0.091
from	189,451	18	0.51	0.091	we	80,245	43	0.22	0.093
as	181,714	19	0.49	0.093	more	76,388	44	0.21	0.090
be	157,300	20	0.42	0.084	after	75,165	45	0.20	0.091
were	153,913	21	0.41	0.087	us	72,045	46	0.19	0.089
an	152,576	22	0.41	0.090	percent	71,956	47	0.19	0.091
have	149,749	23	0.40	0.092	up	71,082	48	0.19	0.092
his	142,285	24	0.38	0.092	one	70,266	49	0.19	0.092
but	140,880	25	0.38	0.094	people	68,988	50	0.19	0.093

Croft et al., "Search Engines, Information Retrieval in Practice"

Heaps' law



$$vocab = const \cdot words^{\beta}$$

Croft et al., "Search Engines, Information Retrieval in Practice"

Outline

Data processing summary

- Lexical analysis
- Stop-word removal
- Stemming
  - Algorithmic (Porter stemmer)
  - Dictionary-based
  - Hybrid (Krovetz stemmer)
- Dealing with phrases
- Zipf's and Heaps' laws

Materials

- Croft et al., Chapter 4
- Manning et al., Chapter 2.2

Data storage methods

- File
- File system
- Database
- **Index**

Outline

Example

S_1 Tropical fish include fish found in tropical environments around the world, including both freshwater and salt water species.

S_2 Fishkeepers often use the term tropical fish to refer only those requiring fresh water, with saltwater tropical fish referred to as marine fish.

S_3 Tropical fish are popular aquarium fish, due to their often bright coloration.

S_4 In freshwater fish, this coloration typically derives from iridescence, while salt water fish are generally pigmented.

Croft et al., "Search Engines, Information Retrieval in Practice"

Outline

Document identifiers

and	1				only	2			
aquarium	3				pigmented	4			
are	3	4			popular	3			
around	1				refer	2			
as	2				referred	2			
both	1				requiring	2			
bright	3				salt	1	4		
coloration	3	4			saltwater	2			
derives	4				species	1			
due	3				term	2			
environments	1				the	1	2		
fish	1	2	3	4	their	3			
fishkeepers	2				this	4			
found	1				those	2			
fresh	2				to	2	3		
freshwater	1	4			tropical	1	2	3	
from	4				typically	4			
generally	4				use	2			
in	1	4			water	1	2	4	
include	1				while	4			
including	1				with	2			
iridescence	4				world	1			
marine	2								
often	2	3							

Croft et al., "Search Engines, Information Retrieval in Practice"

Word counts

and	1:1			
aquarium	3:1			
are	3:1	4:1		
around	1:1			
as	2:1			
both	1:1			
bright	3:1			
coloration	3:1	4:1		
derives	4:1			
due	3:1			
environments	1:1			
fish	1:2	2:3	3:2	4:2
fishkeepers	2:1			
found	1:1			
fresh	2:1			
freshwater	1:1	4:1		
from	4:1			
generally	4:1			
in	1:1	4:1		
include	1:1			
including	1:1			
iridescence	4:1			
marine	2:1			
often	2:1	3:1		
only	2:1			
pigmented	4:1			
popular	3:1			
refer	2:1			
referred	2:1			
requiring	2:1			
salt	1:1	4:1		
saltwater	2:1			
species	1:1			
term	2:1			
the	1:1	2:1		
their	3:1			
this	4:1			
those	2:1			
to	2:2	3:1		
tropical	1:2	2:2	3:1	
typically	4:1			
use	2:1			
water	1:1	2:1	4:1	
while	4:1			
with	2:1			
world	1:1			

Croft et al., "Search Engines, Information Retrieval in Practice"

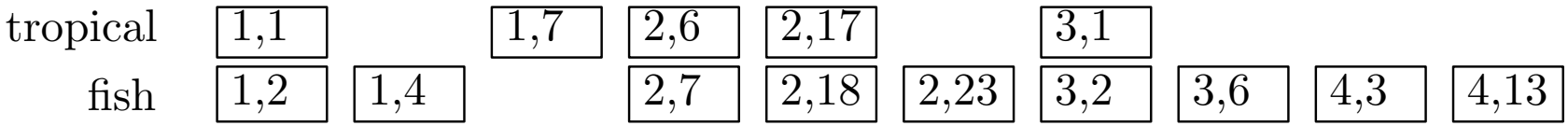
Word positions

and	1,15				
aquarium	3,5				
are	3,3	4,14			
around	1,9				
as	2,21				
both	1,13				
bright	3,11				
coloration	3,12	4,5			
derives	4,7				
due	3,7				
environments	1,8				
fish	1,2	1,4	2,7	2,18	2,23
			3,2	3,6	4,3
			4,13		
fishkeepers	2,1				
found	1,5				
fresh	2,13				
freshwater	1,14	4,2			
from	4,8				
generally	4,15				
in	1,6	4,1			
include	1,3				
including	1,12				
iridescence	4,9				

marine	2,22				
often	2,2	3,10			
only	2,10				
pigmented	4,16				
popular	3,4				
refer	2,9				
referred	2,19				
requiring	2,12				
salt	1,16	4,11			
saltwater	2,16				
species	1,18				
term	2,5				
the	1,10	2,4			
their	3,9				
this	4,4				
those	2,11				
to	2,8	2,20	3,8		
tropical	1,1	1,7	2,6	2,17	3,1
typically	4,6				
use	2,3				
water	1,17	2,14	4,12		
while	4,10				
with	2,15				
world	1,11				

Croft et al., "Search Engines, Information Retrieval in Practice"

Using positions to deal with phrases



Croft et al., "Search Engines, Information Retrieval in Practice"

Index types summary

- Documents
- Documents, counts
- Documents, counts, positions

Outline

Example

D1 **To be**, or not to be...

D2 ...**to die**, to sleep no more...

to	→	D1, D2
be	→	D1
or	→	D1
not	→	D1
die	→	D2
sleep	→	D2
no	→	D2
more	→	D2

Simple indexer

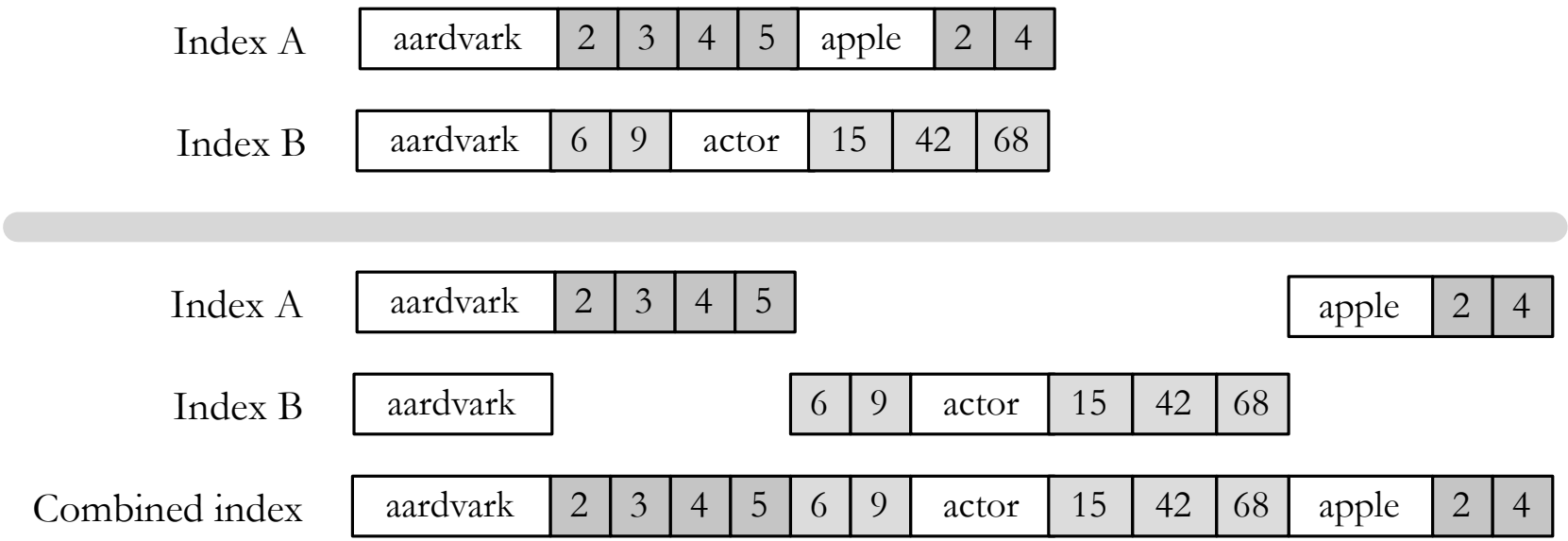
```
procedure BuildIndex(D)
    I ← HashTable()
    n ← 0
    for all documents d ∈ D do
        n ← n + 1
        T ← Parse(d)
        Remove duplicates from T
        for all tokens t ∈ T do
            if I_t ∉ I then
                I_t ← Array()
            end if
            I_t.append(n)
        end for
    end for
    return I
end procedure
```

Croft et al., "Search Engines, Information Retrieval in Practice"

What are the problems with this simple indexer?

1. In-memory
   - Index merging
2. Single-threaded
   - Distributed indexing

Index merging

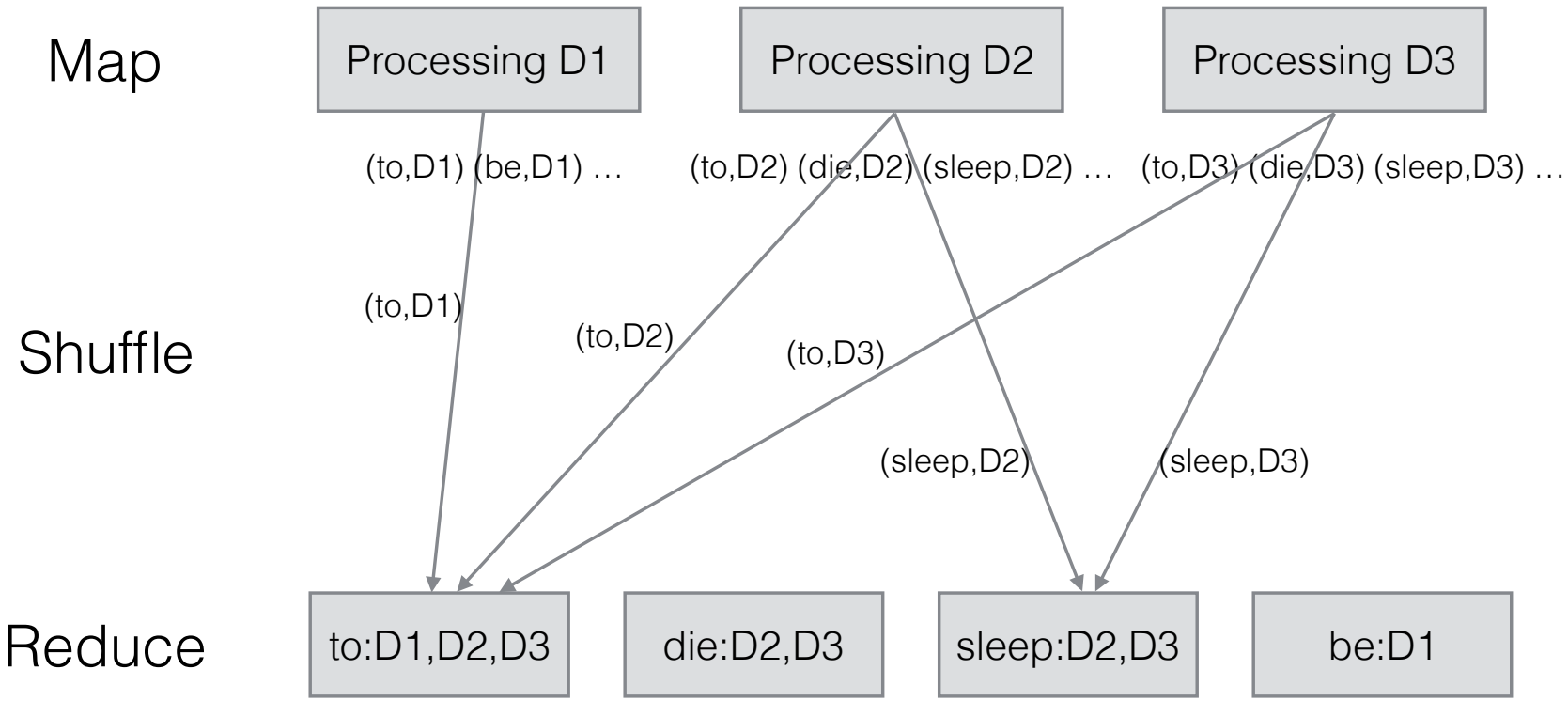


Croft et al., "Search Engines, Information Retrieval in Practice"

Aardvark

Picture taken from `https://en.wikipedia.org/wiki/Aardvark`

Distributed indexing (MapReduce)



Distributed indexing (MapReduce)

```
procedure MapDocumentsToPostings(input)
    while not input.done() do
        document ← input.next()
        number ← document.number
        position ← 0
        tokens ← Parse(document)
        for each word w in tokens do
            Emit(w, document:position)
            position = position + 1
        end for
    end while
end procedure
```

```
procedure ReducePostingsToLists(key, values)
    word ← key
    WriteWord(word)
    while not input.done() do
        EncodePosting(values.next())
    end while
end procedure
```

Croft et al., "Search Engines, Information Retrieval in Practice"

Updating an index

- Index merging – when new documents come in large batches
  1. Create a new index
  2. Merge it with the main index
  3. Remove deleted documents while merging
- Results merging – to handle single document updates
  1. Keep a small in-memory index with new documents
  2. Keep a list of deleted documents
  3. Merge results from the main index and the in-memory index
  4. Do not show results which are in the "deleted list"

Outline

Query processing

- Single-word queries
- Multiple-word queries
  - **AND**
  - OR
  - NOT

and	1					only	2		
aquarium	3					pigmented	4		
are	3	4				popular	3		
around	1					refer	2		
as	2					referred	2		
both	1					requiring	2		
bright	3					salt	1	4	
coloration	3	4				saltwater	2		
derives	4					species	1		
due	3					term	2		
environments	1					the	1	2	
fish	1	2	3	4		their	3		
fishkeepers	2					this	4		
found	1					those	2		
fresh	2					to	2	3	
freshwater	1	4				tropical	1	2	3
from	4					typically	4		
generally	4					use	2		
in	1	4				water	1	2	4
include	1					while	4		
including	1					with	2		
iridescence	4					world	1		
marine	2								
often	2	3							

Croft et al., "Search Engines, Information Retrieval in Practice"

Simple intersection

```
INTERSECT(p1, p2)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then ADD(answer, docID(p1))
 5             p1 ← next(p1)
 6             p2 ← next(p2)
 7        else if docID(p1) < docID(p2)
 8                then p1 ← next(p1)
 9                else p2 ← next(p2)
10  return answer
```

Manning et al., "Introduction to Information Retrieval"

Complexity of simple intersection

- What is the complexity of simple intersection for lists of sizes $\{n_1, \ldots, n_k\}$?

$$O(n_1 + n_2 + \ldots + n_k)$$

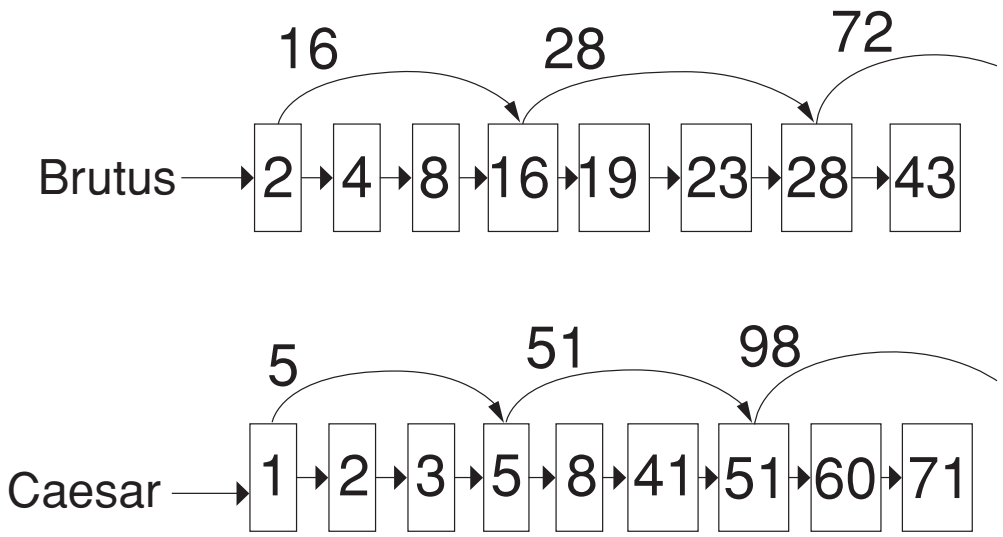
- Heuristic optimization: start with the shortest list
  - Best case:

$$O(k \cdot \min[n_1, \ldots, n_k])$$

  - Worst case:

$$O(n_1 + n_2 + \ldots + n_k)$$

Skip-list optimization



For a list of size P use $\sqrt{P}$ skip-pointers

Manning et al., "Introduction to Information Retrieval"

Skip-list optimization

```
IntersectWithSkips(p1, p2)
 1  answer ← ⟨ ⟩
 2  while p1 ≠ NIL and p2 ≠ NIL
 3  do if docID(p1) = docID(p2)
 4        then Add(answer, docID(p1))
 5             p1 ← next(p1)
 6             p2 ← next(p2)
 7        else if docID(p1) < docID(p2)
 8                then if hasSkip(p1) and (docID(skip(p1)) ≤ docID(p2))
 9                     then while hasSkip(p1) and (docID(skip(p1)) ≤ docID(p2))
10                          do p1 ← skip(p1)
11                     else p1 ← next(p1)
12                else if hasSkip(p2) and (docID(skip(p2)) ≤ docID(p1))
13                     then while hasSkip(p2) and (docID(skip(p2)) ≤ docID(p1))
14                          do p2 ← skip(p2)
15                     else p2 ← next(p2)
16  return answer
```

Manning et al., "Introduction to Information Retrieval"

Outline

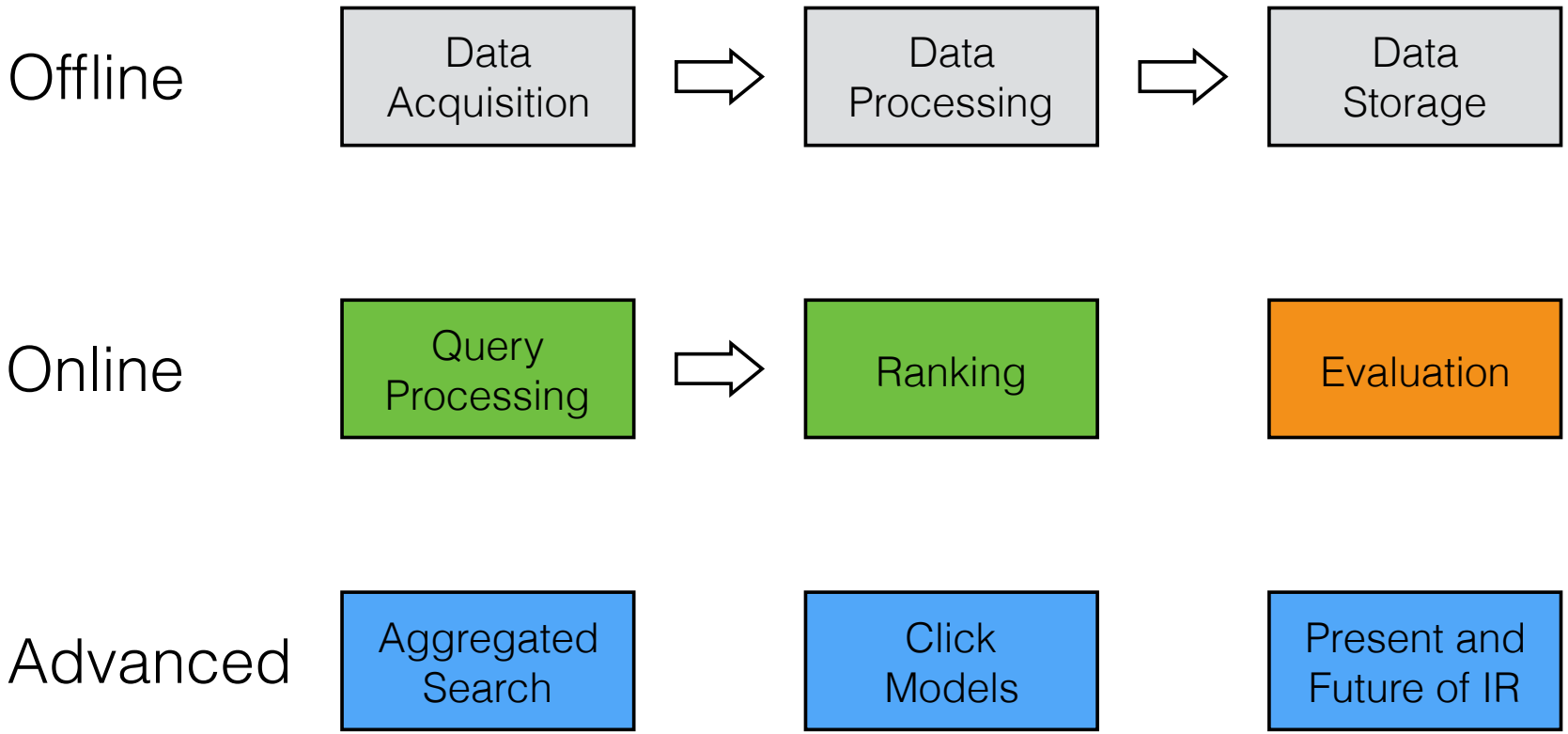
Data storage summary

- Index types
  - Documents
  - Documents, counts
  - Documents, counts, positions
- Index construction
  - Index merging
  - Distributed indexing (MapReduce)
  - Updating an index
- Query processing
  - Boolean operations
  - Skip-list optimization

Materials

- Croft et al., Chapter 5
- Manning et al., Chapters 1.2–1.3, 2.3–2.4

Course overview



See you in October