COMP90042
Web search and text analysis

Workshop Week 3

xudong.han@unimelb.edu.au
https://github.com/HanXudong/COMP90042_Workshops

Review

- Inverted index
- TF*IDF
- BM25

Inverted index

	two	tea	me	you
doc1	0.707	0.707	0	0
doc2	0	0.707	0.353	0.353
doc3	0	0	0.707	0.707

- Query 1: Tea me
- Query 2: Two

$$S_{TF-IDF}(d, Q) = \sum_{t \in Q} tf_{d,t} \times log \frac{N}{df_t}$$

two	1: 0.707;
tea	1:0.707; 2: 0.707
me	2: 0.353; 3:0.707
you	2: 0.353; 3:0.707

This workshop

- Postings list
- Variable Byte Compression
- WAND
- Query expansion
- Relevance feedback

Posting List Compression

Motivations:

- Minimise storage costs
- Fast sequential access
- Support GEQ(x) operation: Return the smallest item in the list that is greater or equal to x

Posting List Compression

Inverted index

the	ids:	25	26	29	…	12345	12347
house	ids:	5213	5234	5454	5591	…	
aeronaut	ids:	251235	251239	251240			

8	10	13	15	18
256	1024	8192	32768	262144

Posting List Compression

the	ids:	25	26	29	…	12345	12347
	gaps:	25	1	3	…	1	2
house	ids:	5213	5234	5454	5591	…	
	gaps:	5213	1	220	137	…	
aeronaut	ids:	251235	251239	251240			
	gaps:	251235	4	1			

Gaps between ids or term frequencies?

Variable Byte Compression

Idea of Variable Byte Compression:

Use variable number of bytes to represent integers. Each byte contains 7 bits “payload” and one continuation bit.

Number	Encoding	
824	00000110	10111000
5	10000101	

Bitwise operators

https://wiki.python.org/moin/BitwiseOperators

The Operators:

- x << y
  Returns x with the bits shifted to the left by y places

- x >> y
  Returns x with the bits shifted to the right by y places.

- x & y
  Does a "bitwise and".

- x | y
  Does a "bitwise or".

- x ^ y
  Does a "bitwise exclusive or".

Variable Byte Compression

Encoding

```
1: function ENCODE(x)
2:     while x >= 128 do
3:         WRITE(x mod 128)
4:         x = x ÷ 128
5:     end while
6:     WRITE(x + 128)
7: end function
```

Decoding

```
1: function DECODE(bytes)
2:     x = 0, s = 0
3:     y = READBYTE(bytes)
4:     while y < 128 do
5:         x = x ^ (y << s)
6:         s = s + 7
7:         y = READBYTE(bytes)
8:     end while
9:     x = x ^ ((y − 128) << s)
10:    return x
11: end function
```

Q: why do we use “ ^ ”?

Variable Byte Compression

Decoding(Q1-c):

Determine the values of integers X and Y that were encoded as the byte sequence [52,34,147,42,197] using the Variable Byte algorithm described in the lecture slides 9/10.

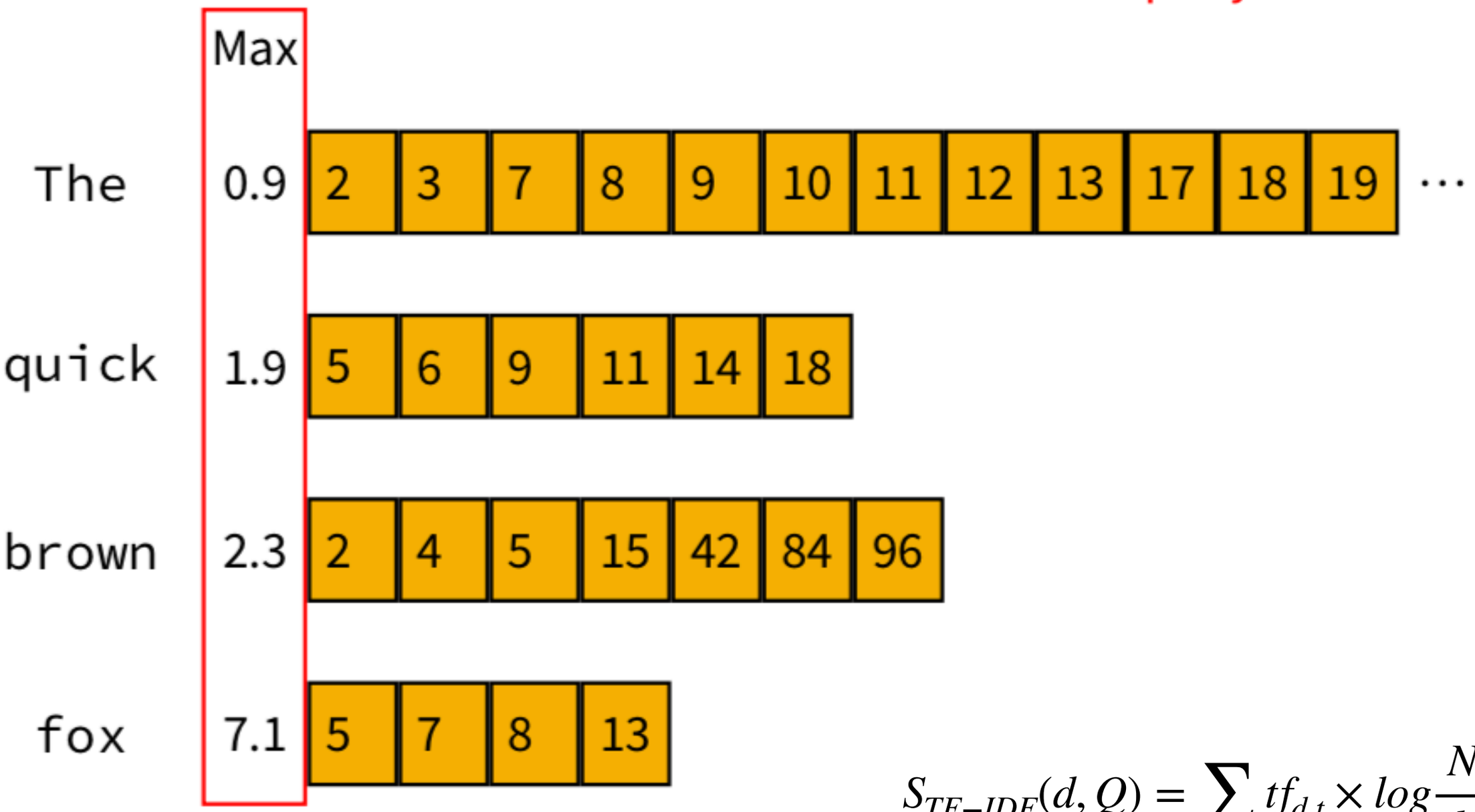
52	00110100
34	00100010
147	10010011
42	00101010
167	11000101

WAND

- **Top K retrieval**
- **Overestimate**

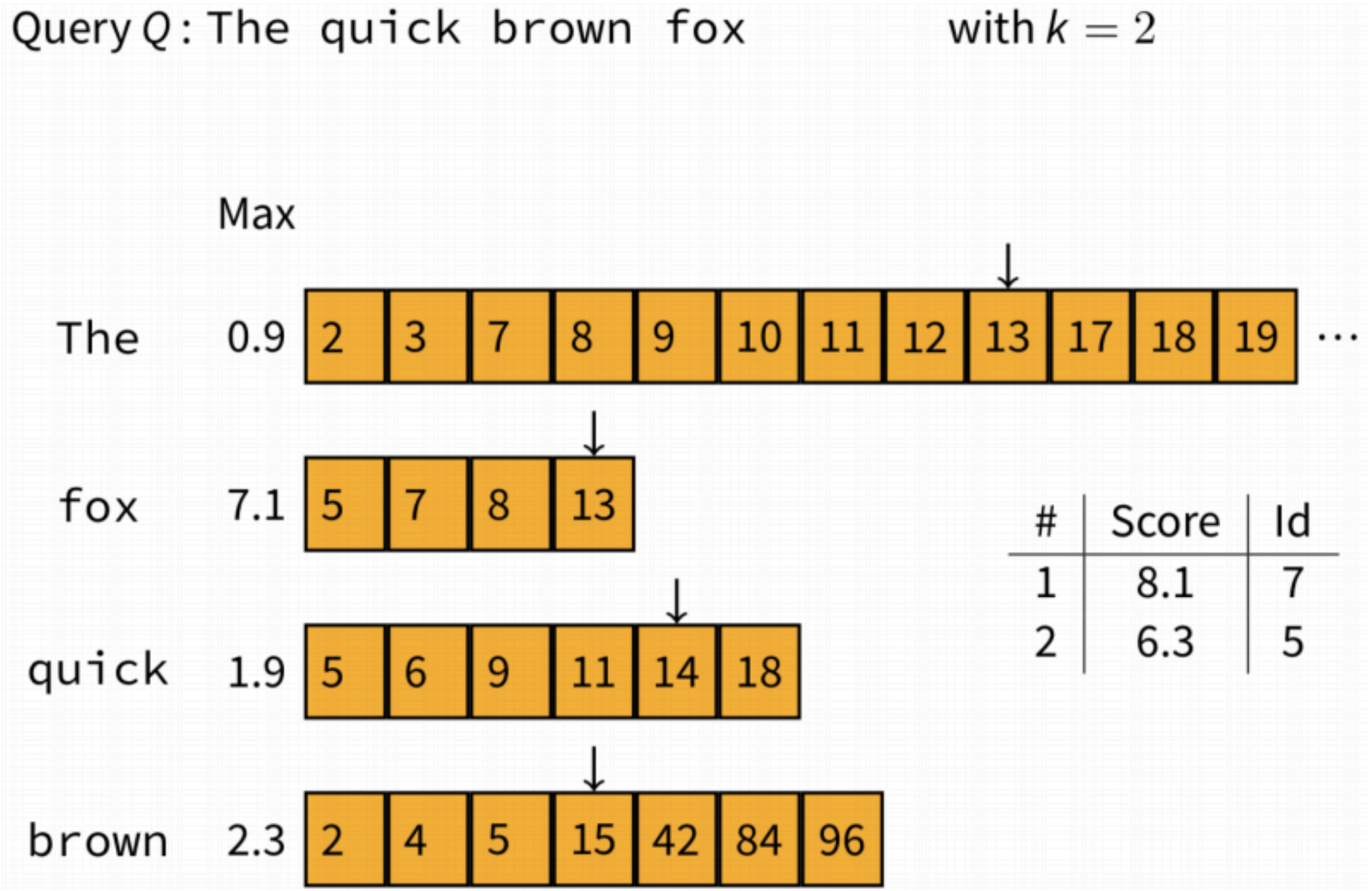
Query Q: `The quick brown fox` with $k = 2$

Maximum Contribution for each query term



$$S_{TF-IDF}(d, Q) = \sum_{t \in Q} tf_{d,t} \times log \frac{N}{df_t}$$

Assume Document 13 has just been evaluated. In the setting below, what is the next document that will be evaluated?



#	Score	Id
1	8.1	7
2	6.3	5

Query Expansion

Q3

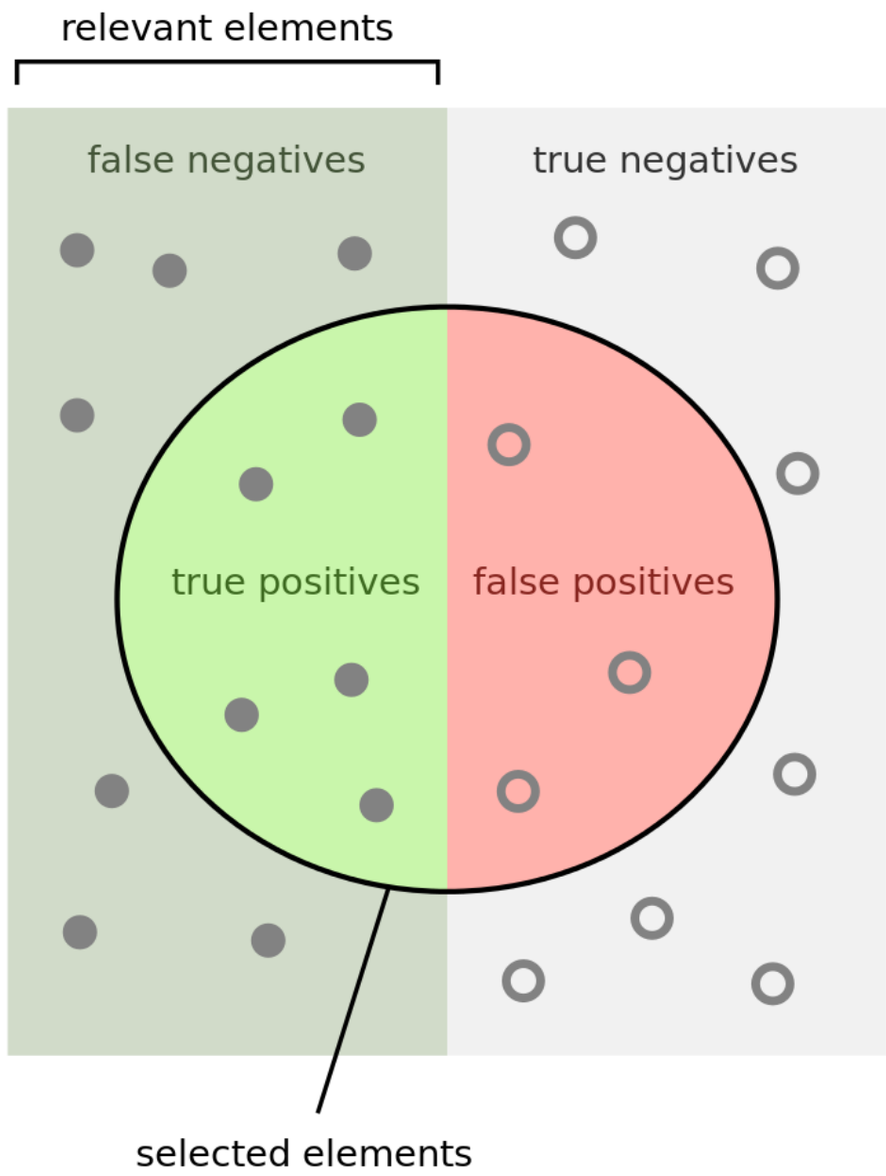
Query expansion increases query recall

How many selected items are relevant?

Precision =

How many relevant items are selected?

Recall =



Recall and Precision

- **Documents: [1, 2, 3, 4, 5, … ,99, 100]**
- **Relevance documents: [1, 3, 5, 7, 9]**
- **Prediction 1: [1, 2, 3, 4, 5, 6, 7, 8, 9]**

$$Recall = \frac{5}{5} \quad Precision = \frac{5}{9}$$

- **Prediction 2: [1, 2, 3, 4, 5, … ,99, 100]**

$$Recall = \frac{5}{5} \quad Precision = \frac{5}{100}$$

- **Prediction 3: [1]**

$$Recall = \frac{1}{5} \quad Precision = \frac{1}{1}$$

Relevance Feedback

Q4

A. **User relevance feedback**
   -E.g. ask users to click

B. **Pseudo relevance feedback**
   -E.g. blink feedback

C. **Indirect relevance feedback**
   -E.g. analysis query click logs to re-rank

Relevance Feedback

Q5 query expansion without relevance feedback

WordNet based query expansion

“Improving Query Expansion Using WordNet”

https://arxiv.org/abs/1309.4938