Search Engine Architecture

3. Indexing and Retrieval

Noted slides adapted from Lin et al.'s Data-Intensive Computing with MapReduce, UMD Spring 2013 with cosmetic changes.

The Central Problem in Search

Searcher

Author

Concepts

Concepts

Query Terms

"tragic love story"

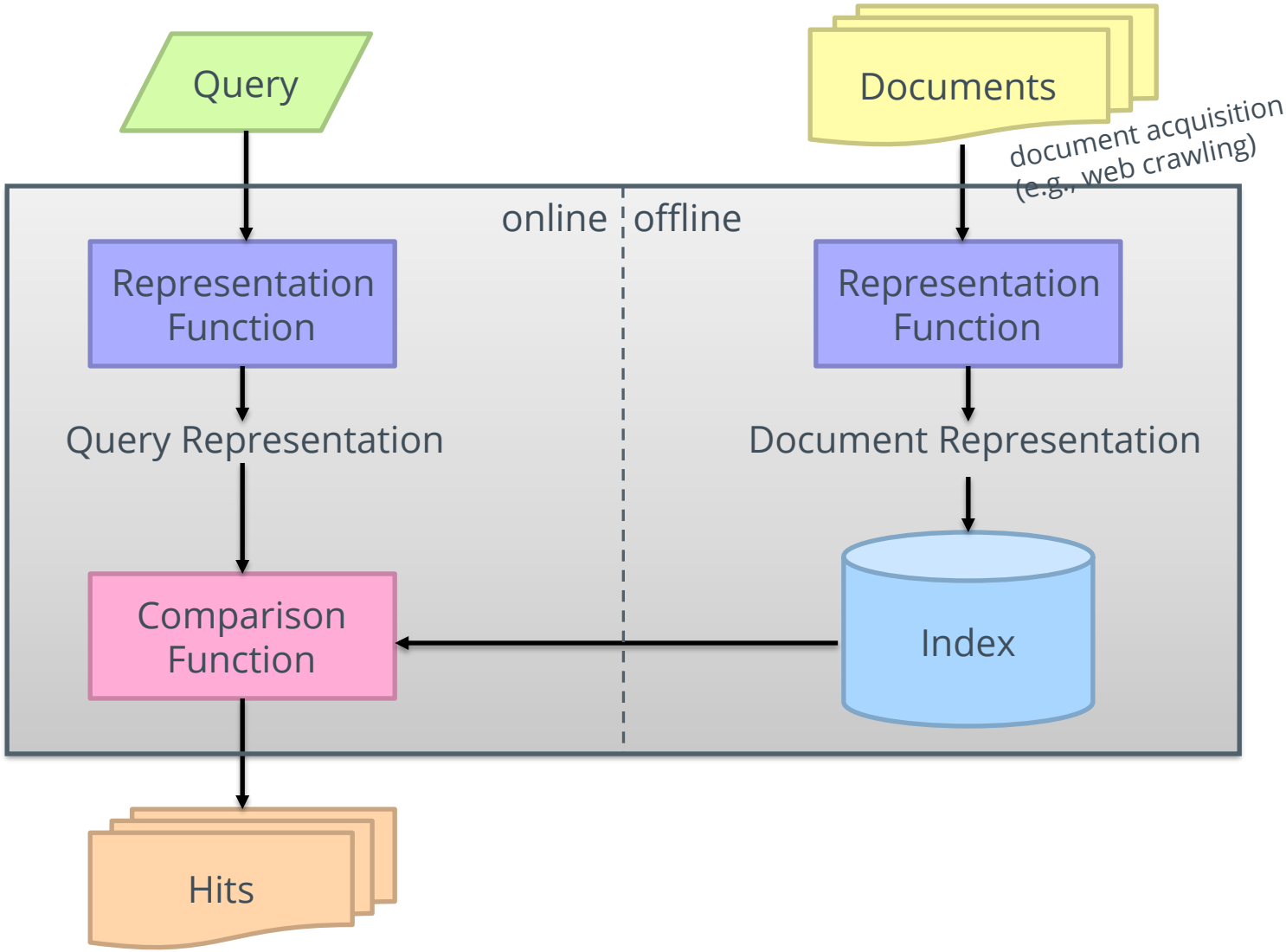
Document Terms

"fateful star-crossed romance"

Do these represent the same concepts?

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Abstract IR Architecture



Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

How do we represent text?

- "Bag of words"
  - Treat all the words in a document as index terms
  - Assign a "weight" to each term based on "importance" (or, in simplest case, presence/absence of word)
  - Disregard order, structure, meaning, etc. of the words
  - Simple, yet effective!
- Assumptions
  - Term occurrence is independent
  - Document relevance is independent
  - "Words" are well-defined

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

What's a word?

天主教教宗若望保祿二世因感冒再度住進醫院。這是他今年第二度因同樣的病因住院。

وقال مارك ريجيف - الناطق باسم الخارجية الإسرائيلية - إن شارون قبل الدعوة وسيقوم للمرة الأولى بزيارة تونس، التي كانت لفترة طويلة المقر الرسمي لمنظمة التحرير الفلسطينية بعد خروجها من لبنان عام 1982.

Выступая в Мещанском суде Москвы экс-глава ЮКОСа заявил не совершал ничего противозаконного, в чем обвиняет его генпрокуратура России.

भारत सरकार ने आर्थकि सर्वेक्षण में वर्तिीय वर्ष 2005-06 में सात फ़ीसदी वकिास दर हासलि करने का आकलन कयिा है और कर सुधार पर ज़ोर दयिा है

日米連合で台頭中国に対処…アーミテージ前副長官提言

조재영 기자= 서울시는 25일 이명박 시장이 `행정중심복합도시" 건설안에 대해 `군대라도 동원해 막고싶은 심정"이라고 말했다는 일부 언론의 보도를 부인했다.

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Sample Document

McDonald's slims down spuds

Fast-food chain to reduce certain types of fat in its french fries with new cooking oil.

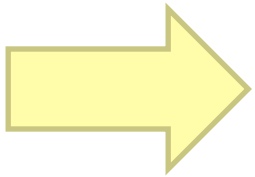
NEW YORK (CNN/Money) - McDonald's Corp. is cutting the amount of "bad" fat in its french fries nearly in half, the fast-food chain said Tuesday as it moves to make all its fried menu items healthier.

But does that mean the popular shoestring fries won't taste the same? The company says no. "It's a win-win for our customers because they are getting the same great french-fry taste along with an even healthier nutrition profile," said Mike Roberts, president of McDonald's USA.

But others are not so sure. McDonald's will not specifically discuss the kind of oil it plans to use, but at least one nutrition expert says playing with the formula could mean a different taste.

Shares of Oak Brook, Ill.-based McDonald's (MCD: down $0.54 to $23.22, Research, Estimates) were lower Tuesday afternoon. It was unclear Tuesday whether competitors Burger King and Wendy's International (WEN: down $0.80 to $34.91, Research, Estimates) would follow suit. Neither company could immediately be reached for comment.

...



"Bag of Words"

14 × McDonalds

12 × fat

11 × fries

8 × new

7 × french

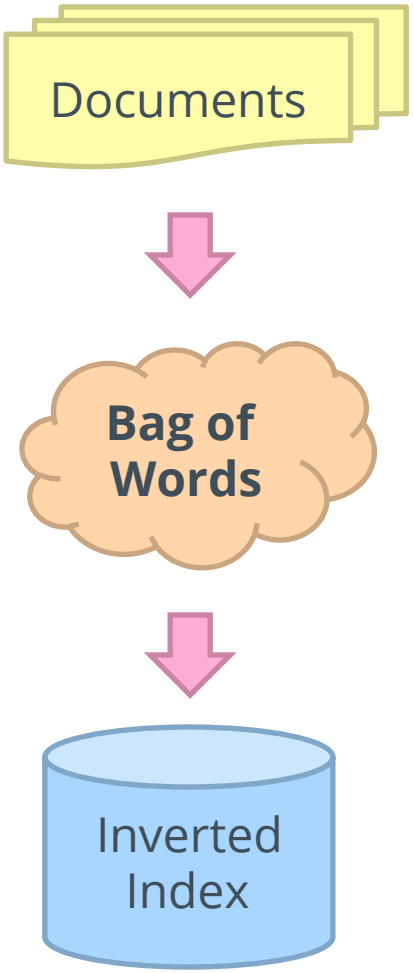
6 × company, said, nutrition

5 × food, oil, percent, reduce, taste, Tuesday

...

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Counting Words…



case folding, tokenization, stopword removal, stemming

Source: Lin et al. Data-Intensive Computing with MapReduce, UMD Spring 2013.

Boolean Retrieval

- Users express queries as a Boolean expression
  - AND, OR, NOT
  - Can be arbitrarily nested
- Retrieval is based on the notion of sets
  - Any given query divides the collection into two sets: retrieved, not-retrieved
  - Pure Boolean systems do not define an ordering of the results

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

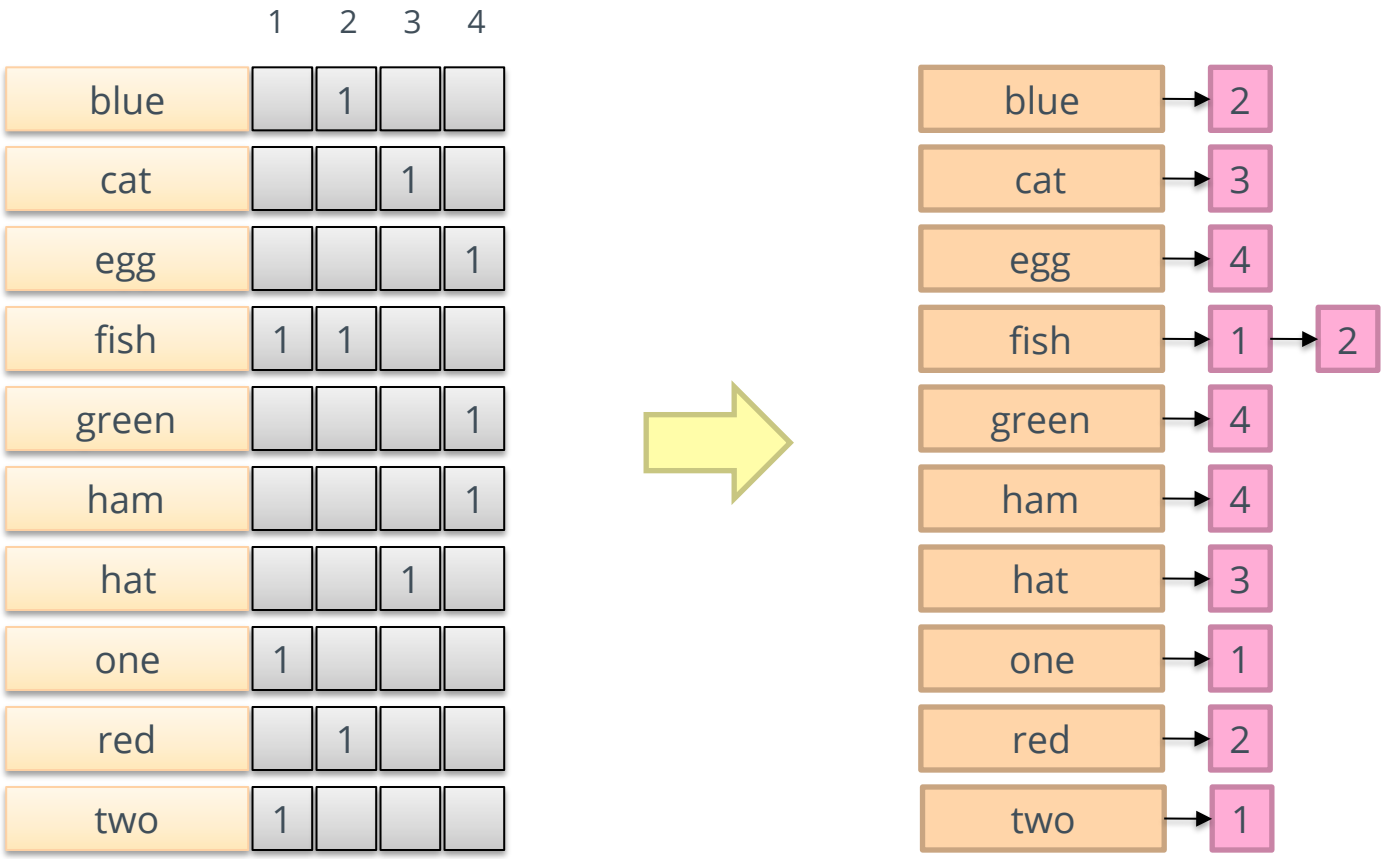
Inverted Index: Boolean Retrieval

Doc 1 one fish, two fish

Doc 2 red fish, blue fish

Doc 3 cat in the hat

Doc 4 green eggs and ham



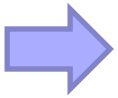
Hybrid of a KV store and a column store?

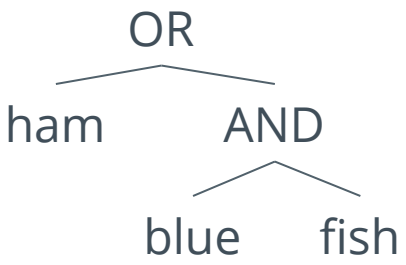
Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Boolean Retrieval

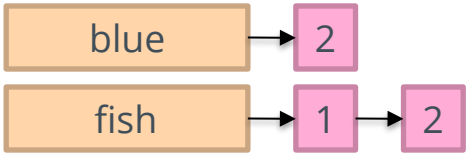
- To execute a Boolean query:
  - Build query syntax tree

(blue AND fish) OR ham





  - For each clause, look up postings



  - Traverse postings and apply Boolean operator
- Efficiency analysis
  - Postings traversal is linear (assuming sorted postings)
  - Start with shortest posting first

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Strengths and Weaknesses

- Strengths
  - Precise, if you know the right strategies
  - Precise, if you have an idea of what you're looking for
  - Implementations are fast and efficient
- Weaknesses
  - Users must learn Boolean logic
  - Boolean logic insufficient to capture the richness of language
  - No control over size of result set: either too many hits or none
  - When do you stop reading? All documents in the result set are considered "equally good"
  - What about partial matches? Documents that "don't quite match" the query may be useful also

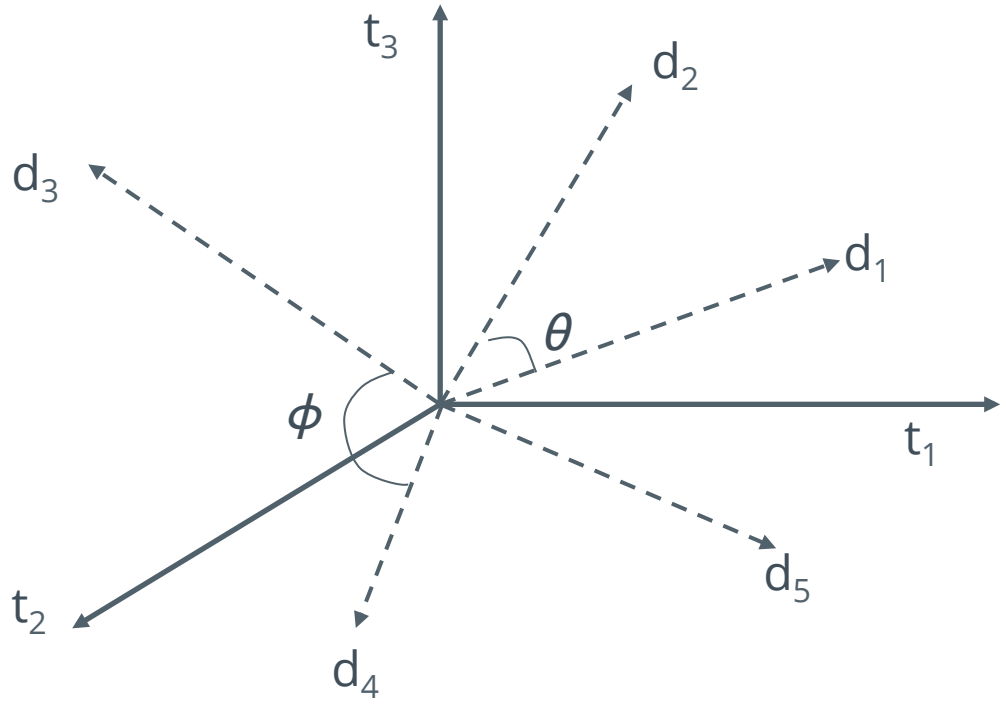
Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Ranked Retrieval

- Order documents by how likely they are to be relevant
  - Estimate relevance(q, d_i)
  - Sort documents by relevance
  - Display sorted results
- User model
  - Present hits one screen at a time, best results first
  - At any point, users can decide to stop looking
- How do we estimate relevance?
  - Intuitively, what factors should relevance(q, d_i) take into account?

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Vector Space Model



Assumption: Documents that are "close together" in vector space "talk about" the same things

Therefore, retrieve documents based on how close the document is to the query (i.e., similarity ~ "closeness")

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Term Weighting

- Term weights consist of two components
  - Local: how important is the term in this document?
  - Global: how important is the term in the collection?
- Here's the intuition:
  - Terms that appear often in a document should get high weights
  - Terms that appear in many documents should get low weights
- How do we capture this mathematically?
  - Term frequency (local)
  - Inverse document frequency (global)

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

TF-IDF Term Weighting

$$w_{i,j} = \mathrm{tf}_{i,j} \cdot \log \frac{N}{n_i}$$

$w_{i,j}$ weight assigned to term *i* in document *j*

$\mathrm{tf}_{i,j}$ number of occurrence of term *i* in document *j*

N number of documents in entire collection

n_i number of documents with term *i*

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

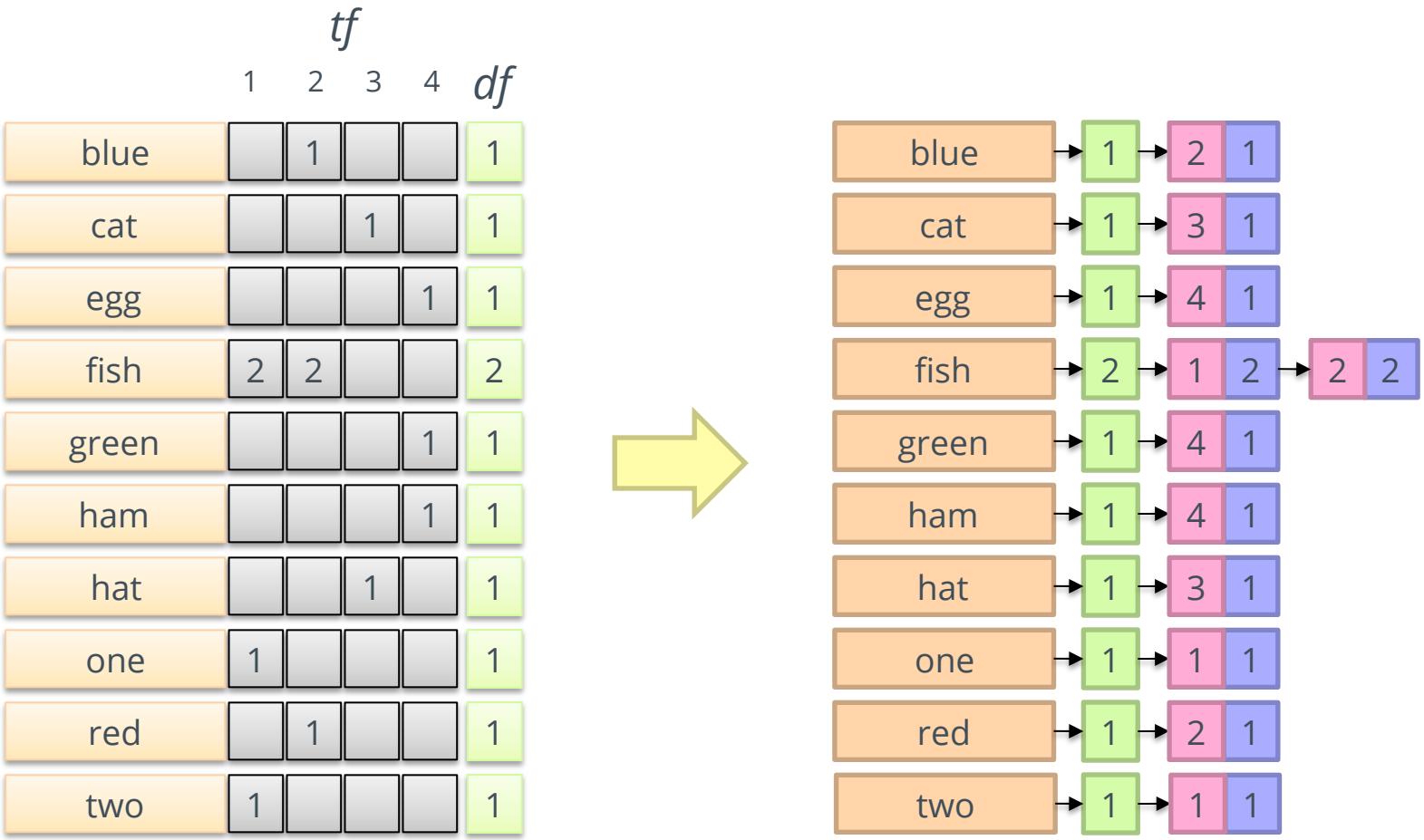
Inverted Index: TF-IDF

Doc 1 one fish, two fish

Doc 2 red fish, blue fish

Doc 3 cat in the hat

Doc 4 green eggs and ham



Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Positional Indexes

- Store term position in postings
- Supports richer queries (e.g., proximity)
- Naturally, leads to larger indexes…

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

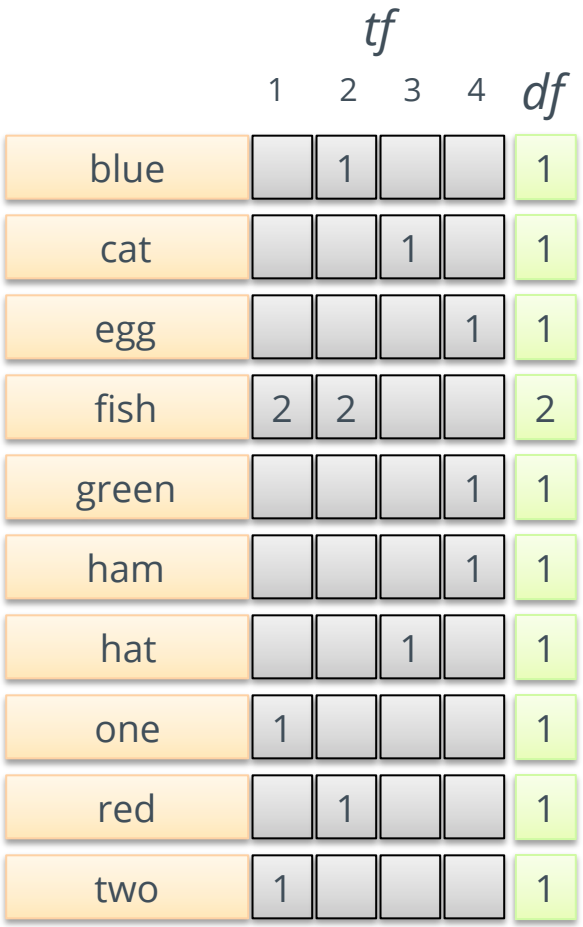
Inverted Index: Positional Information

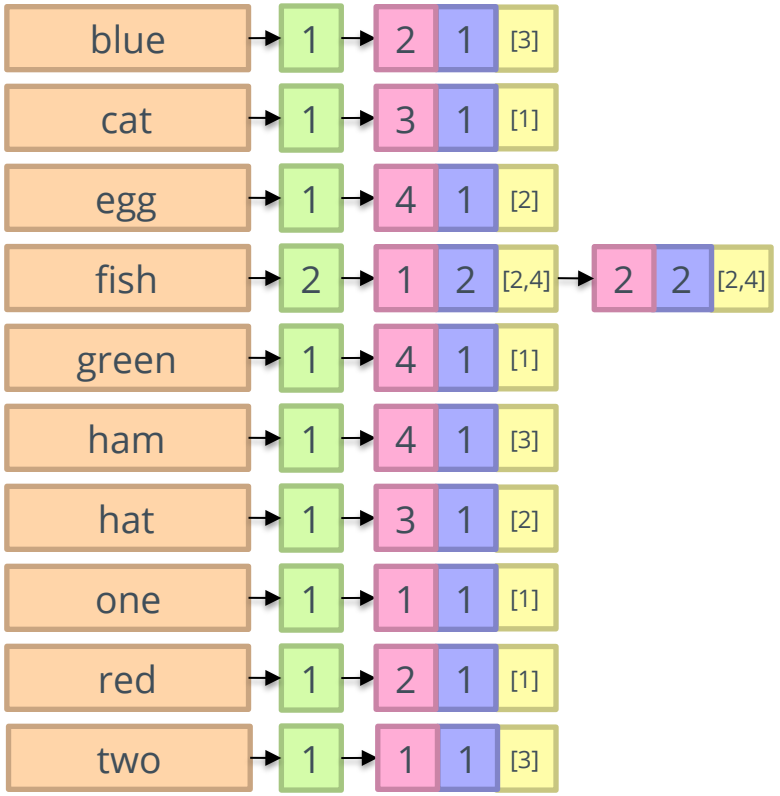
Doc 1 one fish, two fish

Doc 2 red fish, blue fish

Doc 3 cat in the hat

Doc 4 green eggs and ham





Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

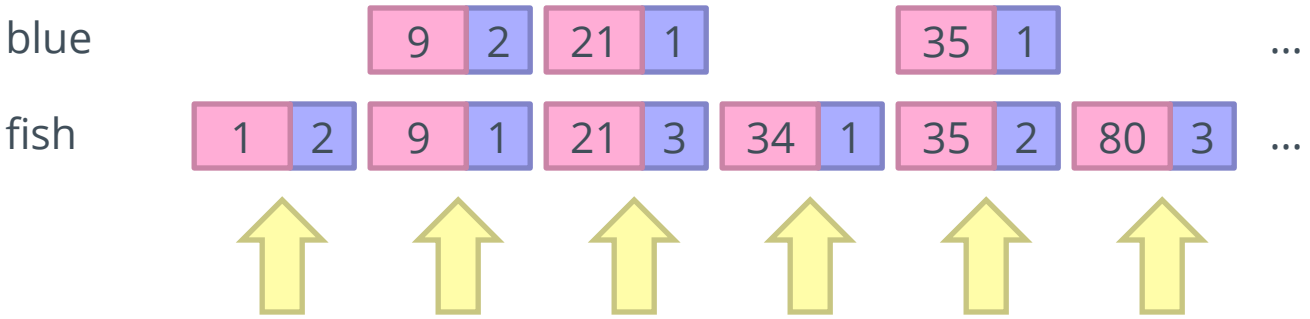
Retrieval in a Nutshell

- Look up postings lists corresponding to query terms
- Traverse postings for each query term
- Store partial query-document scores in accumulators
- Select top *k* results to return

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Retrieval: Document-at-a-Time

- Evaluate documents one at a time (score all query terms)



Accumulators
(e.g. priority queue)

Document score in top k?

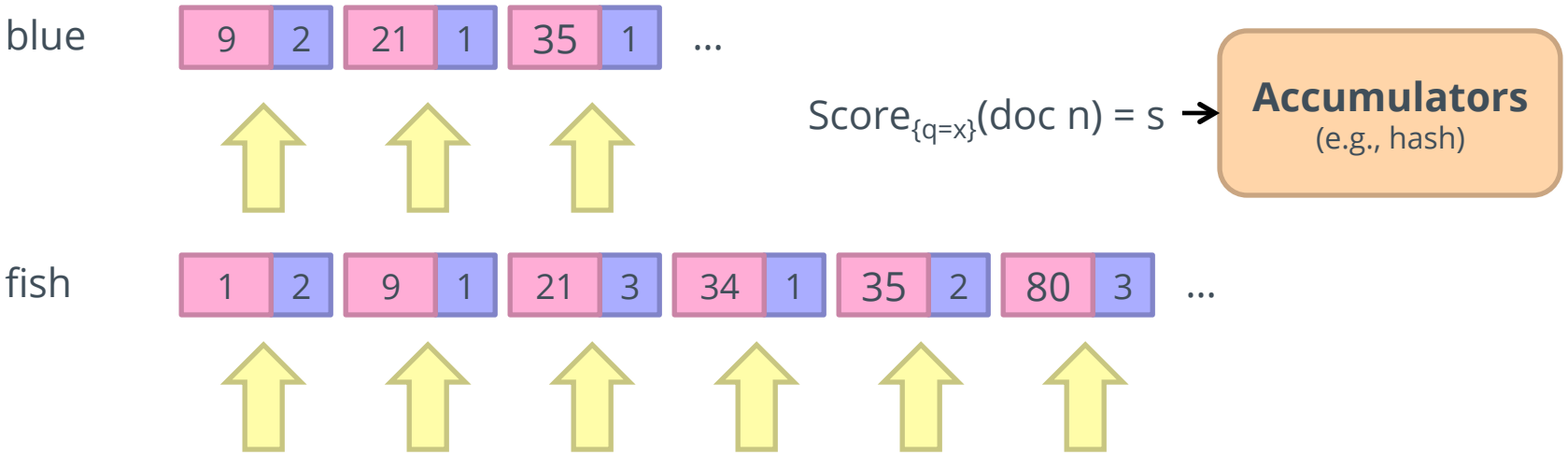
Yes: Insert document score, extract-min if queue too large
No: Do nothing

- Tradeoffs
  - Small memory footprint (good)
  - Must read through all postings (bad), but skipping possible
  - More disk seeks (bad), but blocking possible

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Retrieval: Query-At-A-Time

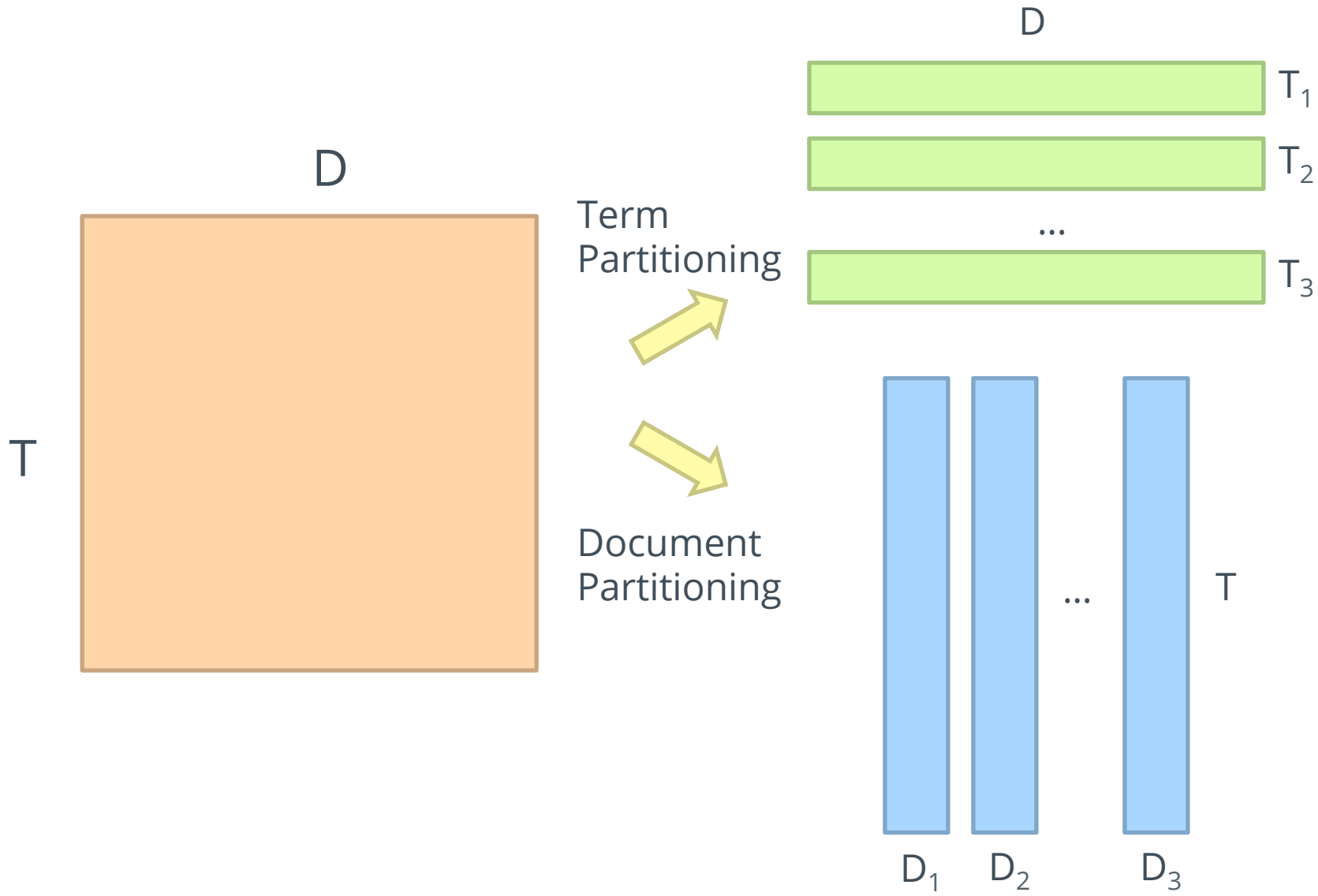
- Evaluate documents one query term at a time
  - Usually, starting from most rare term (often with tf-sorted postings)



- Tradeoffs
  - Early termination heuristics (good)
  - Large memory footprint (bad), but filtering heuristics possible

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Term vs. Document Partitioning



Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Document Partitioning

- Requires query broker
- Lower query latencies
- Hot spots easier to manage
- Supports multi-phase search strategy
  - Start with highest-quality documents, …

Term Partitioning

- Query iterates over terms
- Fewer total disk seeks per query
- Hot spots due to common terms

Indexing vs. Retrieval

- The indexing problem
  - Scalability is critical (to be continued…)
  - Must be relatively fast, but need not be real time
  - Fundamentally a batch operation
  - Incremental updates may or may not be important
  - For the web, crawling is a challenge in itself
- The retrieval problem
  - Must have sub-second response time
  - For the web, need relatively few results

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Postings Encoding

Conceptually:

In Practice:

- Don't encode docnos, encode gaps (or *d*-gaps)
- But it's not obvious that this saves space...

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

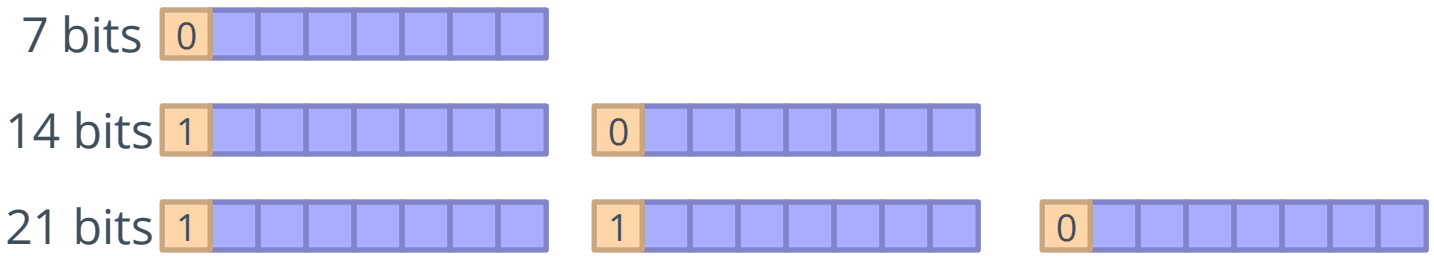
Overview of Index Compression

- Byte-aligned vs. bit-aligned
- Byte-aligned technique
  - VByte
  - Simple9 and variants
  - PForDelta
- Non-parameterized bit-aligned
  - Unary codes
  - γ codes
  - δ codes
- Parameterized bit-aligned
  - Golomb codes (local Bernoulli model)

Want more detail? Read *Managing Gigabytes* by Witten, Moffat, and Bell!

VByte

- Simple idea: use only as many bytes as needed
  - Need to reserve one bit per byte as the “continuation bit”
  - Use remaining bits for encoding value

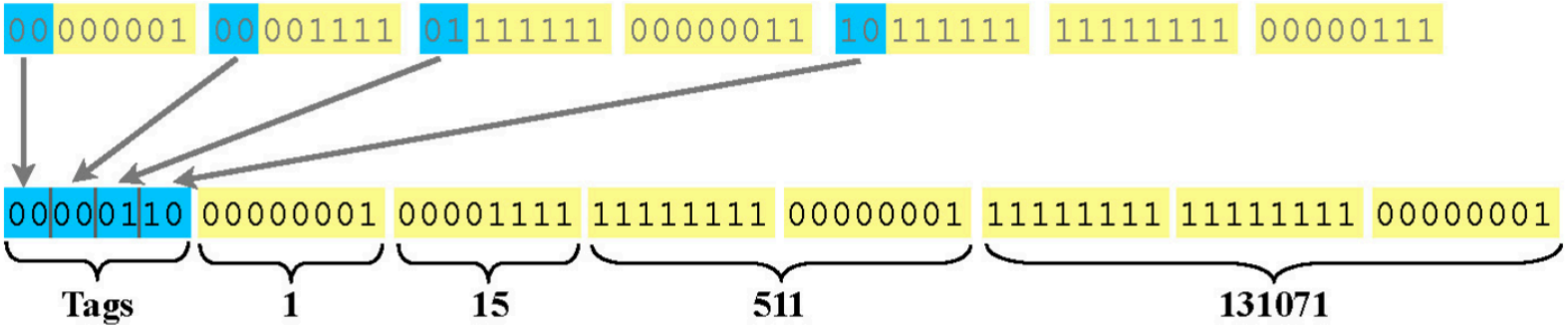


- Works okay, easy to implement…

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Group Varint Encoding

- Idea: encode groups of 4 values in 5-17 bytes
  - Pull out 4 2-bit binary lengths into single byte prefix



- Decode: Load prefix byte and use value to lookup in 256-entry table:

```
...
00 00 01 10 → Offsets: +1,+2,+3,+5; Masks: ff, ff, ffff, ffffff
...
```

- Much faster than alternatives:
  - 7-bit-per-byte varint: decode ~180M numbers/second
  - 30-bit Varint w/ 2-bit length: decode ~240M numbers/second
  - Group varint: decode ~400M numbers/second

Google

Source: Dean. "Challenges in Building Large-Scale Information Retrieval Systems" WSDM 2009.

Unary Codes

- $x \geq 1$ is coded as x-1 one bits, followed by 1 zero bit
  - 3 = 110
  - 4 = 1110
- Great for small numbers… horrible for large numbers
  - Overly-biased for very small gaps

Watch out! Slightly different definitions in different textbooks

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

γ codes

- $x \geq 1$ is coded in two parts: length and offset
  - Start with binary encoded, remove highest-order bit = offset
  - Length is number of binary digits, encoded in unary code
  - Concatenate length + offset codes
- Example: 9 in binary is 1001
  - Offset = 001
  - Length = 4, in unary code = 1110
  - γ code = 1110:001
- Analysis
  - Offset = $\lfloor \log x \rfloor$
  - Length = $\lfloor \log x \rfloor + 1$
  - Total = $2 \lfloor \log x \rfloor + 1$

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

δ codes

- Similar to γ codes, except that length is encoded in γ code
- Example: 9 in binary is 1001
  - Offset = 001
  - Length = 4, in γ code = 11000
  - δ code = 11000:001
- γ codes = more compact for smaller numbers
  δ codes = more compact for larger numbers

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Golomb Codes

- $x \geq 1$, parameter b:
  - $q + 1$ in unary, where $q = \lfloor (x - 1) / b \rfloor$
  - r in binary, where $r = x - qb - 1$, in $\lfloor \log b \rfloor$ or $\lceil \log b \rceil$ bits
- Example:
  - $b = 3$, $r = 0, 1, 2$ (0, 10, 11)
  - $b = 6$, $r = 0, 1, 2, 3, 4, 5$ (00, 01, 100, 101, 110, 111)
  - $x = 9$, $b = 3$: $q = 2$, $r = 2$, code = 110:11
  - $x = 9$, $b = 6$: $q = 1$, $r = 2$, code = 10:100
- Optimal $b \approx 0.69$ (N/df)
  - Different b for every term!

Comparison of Coding Schemes

	Unary	γ	δ	Golomb	
				b=3	b=6
1	0	0	0	0:0	0:00
2	10	10:0	100:0	0:10	0:01
3	110	10:1	100:1	0:11	0:100
4	1110	110:00	101:00	10:0	0:101
5	11110	110:01	101:01	10:10	0:110
6	111110	110:10	101:10	10:11	0:111
7	1111110	110:11	101:11	110:0	10:00
8	11111110	1110:000	11000:000	110:10	10:01
9	111111110	1110:001	11000:001	110:11	10:100
10	1111111110	1110:010	11000:010	1110:0	10:101

Source: Lin et al. Big Data Infrastructure, UMD Spring 2015.

Index Compression: Performance

Comparison of Index Size (bits per pointer)

	Bible	TREC
Unary	262	1918
Binary	15	20
γ	6.51	6.63
δ	6.23	6.38
Golomb	6.09	5.84

← One of the best techniques

Bible: King James version of the Bible; 31,101 verses (4.3 MB)
TREC: TREC disks 1+2; 741,856 docs (2070 MB)

Witten, Moffat, Bell, Managing Gigabytes (1999) via Lin et al. Big Data Infrastructure, UMD Spring 2015.

Questions?