Space- and Time-Efficient Data Structures for Massive Datasets

Giulio Ermanno Pibiri

Referee
Daniel Lemire

Supervisor
Rossano Venturini

Referee
Simon Gog

Department of Computer Science
University of Pisa

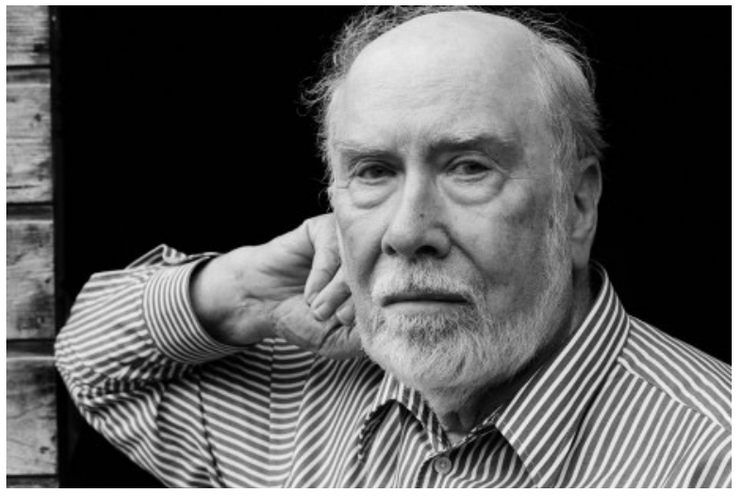
08/03/2019

Evidence

The increase of data and, hence, information does **not** scale with technology.

Evidence

The increase of data and, hence, information does **not** scale with technology.

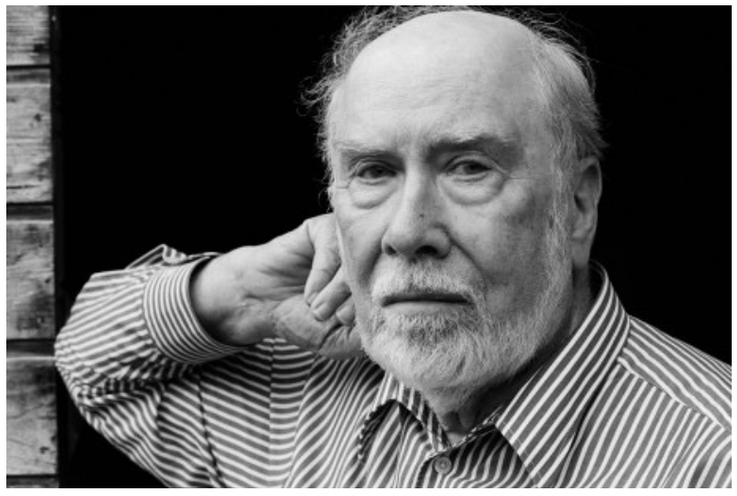


“Software is getting slower more rapidly than hardware becomes faster.”

Niklaus Wirth, A Plea for Lean Software

Evidence

The increase of data and, hence, information does **not** scale with technology.

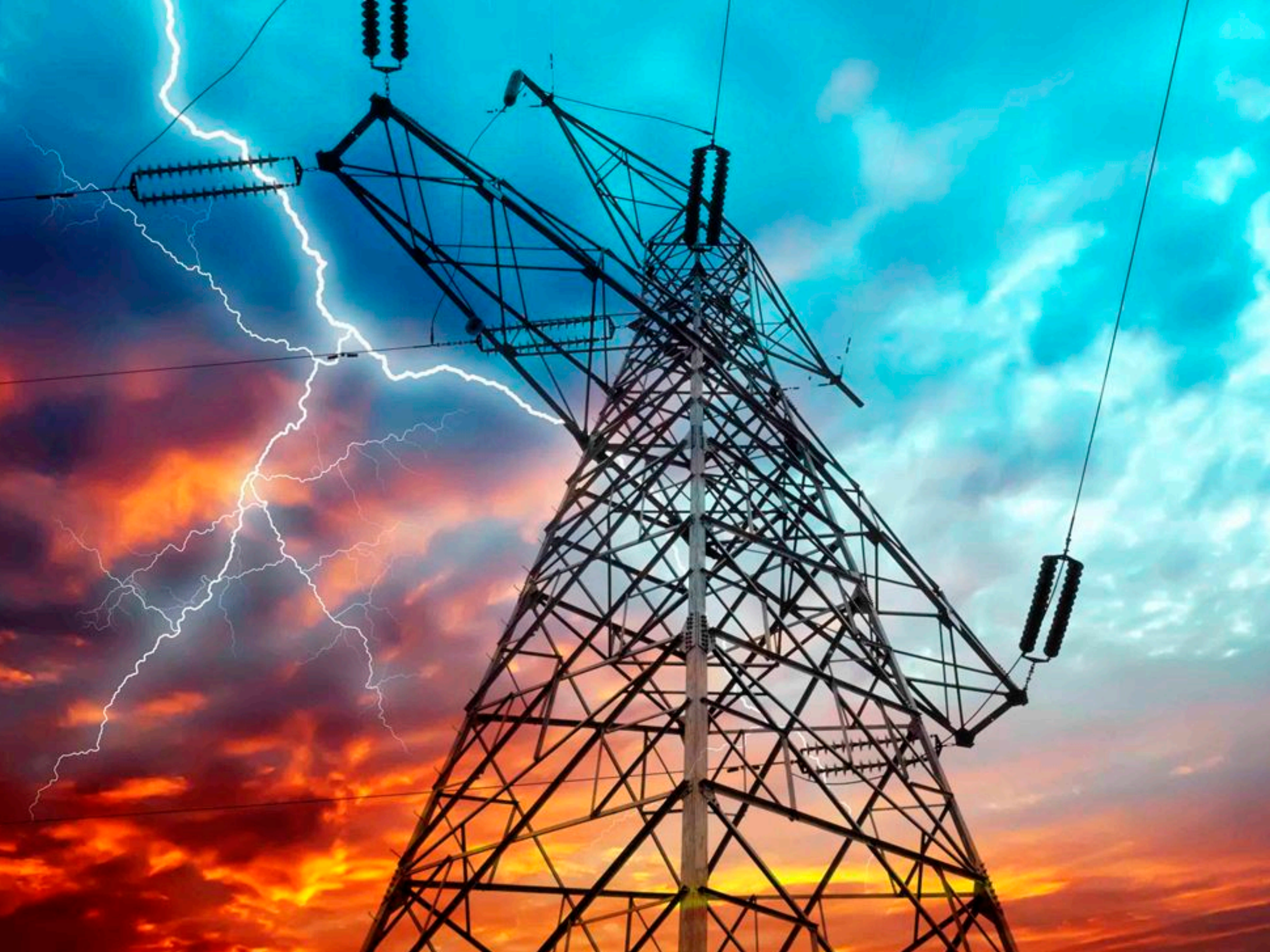


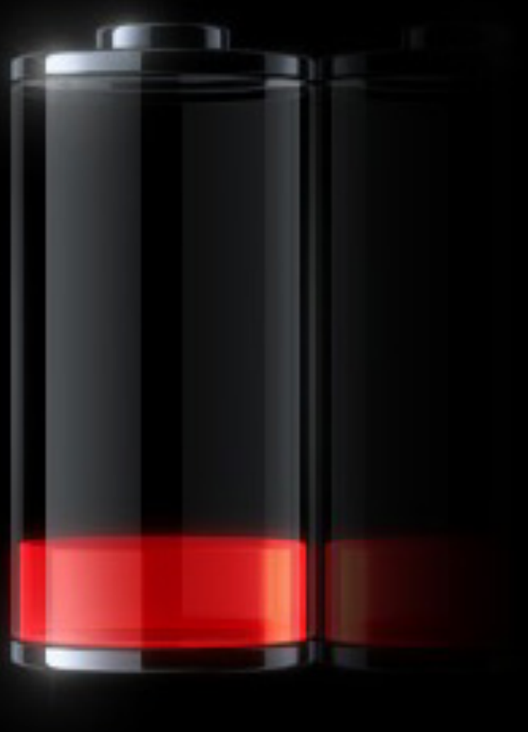
“Software is getting slower more rapidly than hardware becomes faster.”

Niklaus Wirth, A Plea for Lean Software

Even more relevant today!


Statistics
Progress
Growth & Decline
Statistical
Analysis
70%





Achieved results

Clustered Elias-Fano Indexes **Journal paper**

Giulio Ermanno Pibiri and Rossano Venturini
ACM Transactions on Information Systems (TOIS)
Full paper, 34 pages, 2017.

Dynamic Elias-Fano Representation **Conference paper**

Giulio Ermanno Pibiri and Rossano Venturini
Annual Symposium on Combinatorial Pattern Matching (CPM)
Full paper, 14 pages, 2017.

On Optimally Partitioning Variable-Byte Codes **Journal paper**

Giulio Ermanno Pibiri and Rossano Venturini
IEEE Transactions on Knowledge and Data Engineering (TKDE). To appear.
Full paper, 12 pages, 2019.

Fast Dictionary-based Compression for Inverted Indexes **Conference paper**

Giulio Ermanno Pibiri, Matthias Petri and Alistair Moffat
ACM Conference on Web Search and Data Mining (WSDM)
Full paper, 9 pages, 2019.

Efficient Data Structures for Massive N-Gram Datasets **Conference paper**

Giulio Ermanno Pibiri and Rossano Venturini
ACM Conference on Research and Development in Information Retrieval (SIGIR)
Full paper, 10 pages, 2017.

Handling Massive N-Gram Datasets Efficiently **Journal paper**

Giulio Ermanno Pibiri and Rossano Venturini
ACM Transactions on Information Systems (TOIS). To appear.
Full paper, 41 pages, 2019.

Achieved results

integer sequences

Clustered Elias-Fano Indexes — Journal paper

Giulio Ermanno Pibiri and Rossano Venturini
ACM Transactions on Information Systems (TOIS)
Full paper, 34 pages, 2017.

Dynamic Elias-Fano Representation — Conference paper

Giulio Ermanno Pibiri and Rossano Venturini
Annual Symposium on Combinatorial Pattern Matching (CPM)
Full paper, 14 pages, 2017.

On Optimally Partitioning Variable-Byte Codes — Journal paper

Giulio Ermanno Pibiri and Rossano Venturini
IEEE Transactions on Knowledge and Data Engineering (TKDE). To appear.
Full paper, 12 pages, 2019.

Fast Dictionary-based Compression for Inverted Indexes — Conference paper

Giulio Ermanno Pibiri, Matthias Petri and Alistair Moffat
ACM Conference on Web Search and Data Mining (WSDM)
Full paper, 9 pages, 2019.

Efficient Data Structures for Massive N-Gram Datasets — Conference paper

Giulio Ermanno Pibiri and Rossano Venturini
ACM Conference on Research and Development in Information Retrieval (SIGIR)
Full paper, 10 pages, 2017.

Handling Massive N-Gram Datasets Efficiently — Journal paper

Giulio Ermanno Pibiri and Rossano Venturini
ACM Transactions on Information Systems (TOIS). To appear.
Full paper, 41 pages, 2019.

Achieved results

integer sequences

Clustered Elias-Fano Indexes

Journal paper

Giulio Ermanno Pibiri and Rossano Venturini
ACM Transactions on Information Systems (TOIS)
Full paper, 34 pages, 2017.

Dynamic Elias-Fano Representation

Conference paper

Giulio Ermanno Pibiri and Rossano Venturini
Annual Symposium on Combinatorial Pattern Matching (CPM)
Full paper, 14 pages, 2017.

On Optimally Partitioning Variable-Byte Codes

Journal paper

Giulio Ermanno Pibiri and Rossano Venturini
IEEE Transactions on Knowledge and Data Engineering (TKDE). To appear.
Full paper, 12 pages, 2019.

Fast Dictionary-based Compression for Inverted Indexes

Giulio Ermanno Pibiri, Matthias Petri and Alistair Moffat
ACM Conference on Web Search and Data Mining (WSDM)
Full paper, 9 pages, 2019.

Conference paper

short strings

Efficient Data Structures for Massive N-Gram Datasets

Conference paper

Giulio Ermanno Pibiri and Rossano Venturini
ACM Conference on Research and Development in Information Retrieval (SIGIR)
Full paper, 10 pages, 2017.

Handling Massive N-Gram Datasets Efficiently

Journal paper

Giulio Ermanno Pibiri and Rossano Venturini
ACM Transactions on Information Systems (TOIS). To appear.
Full paper, 41 pages, 2019.

Problem 1

Consider a sorted integer sequence.

Problem 1

Consider a sorted integer sequence.

How to represent it as a bit-vector where each original integer is uniquely-decodable, using **as few as possible** bits?

How to maintain **fast decompression speed**?

Ubiquity

Inverted indexes



Databases



E-Commerce


Graph compression



Semantic data

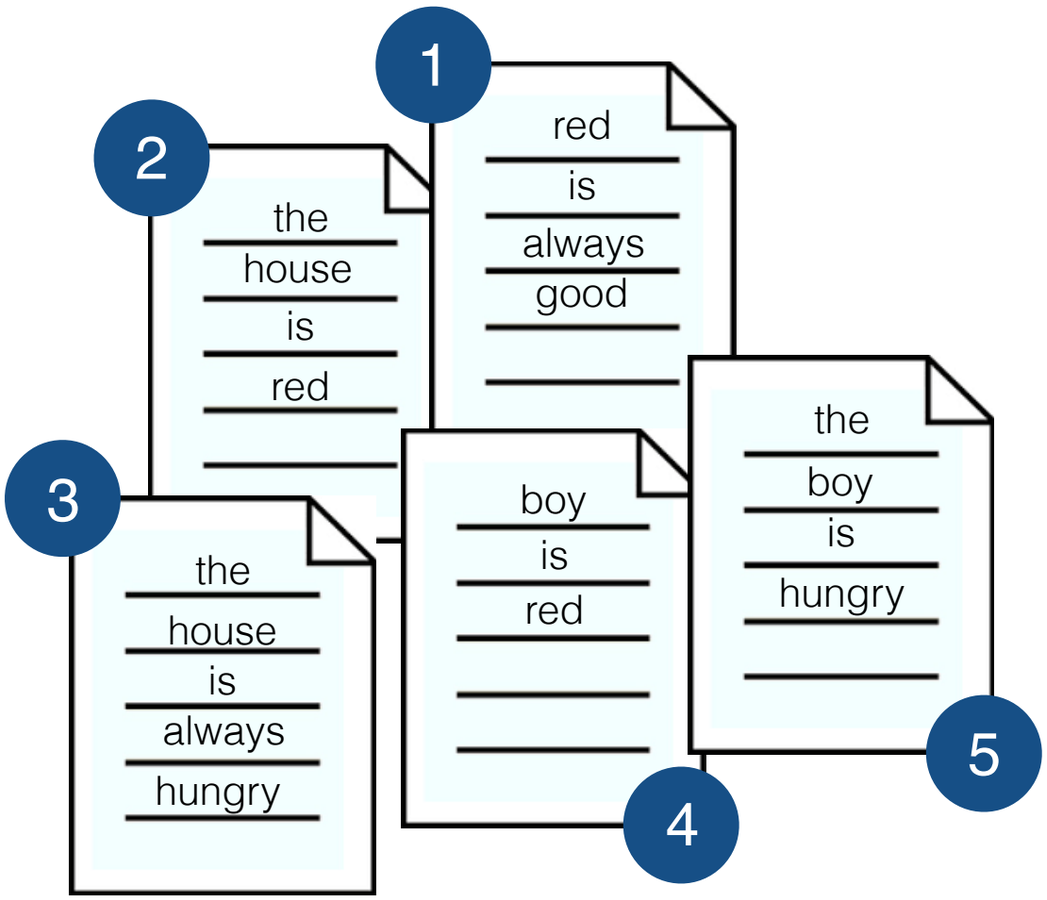

Geo-spatial data


Inverted indexes

The inverted index is the *de-facto* data structure at the basis of every large-scale retrieval system.

Inverted indexes

The inverted index is the *de-facto* data structure at the basis of every large-scale retrieval system.

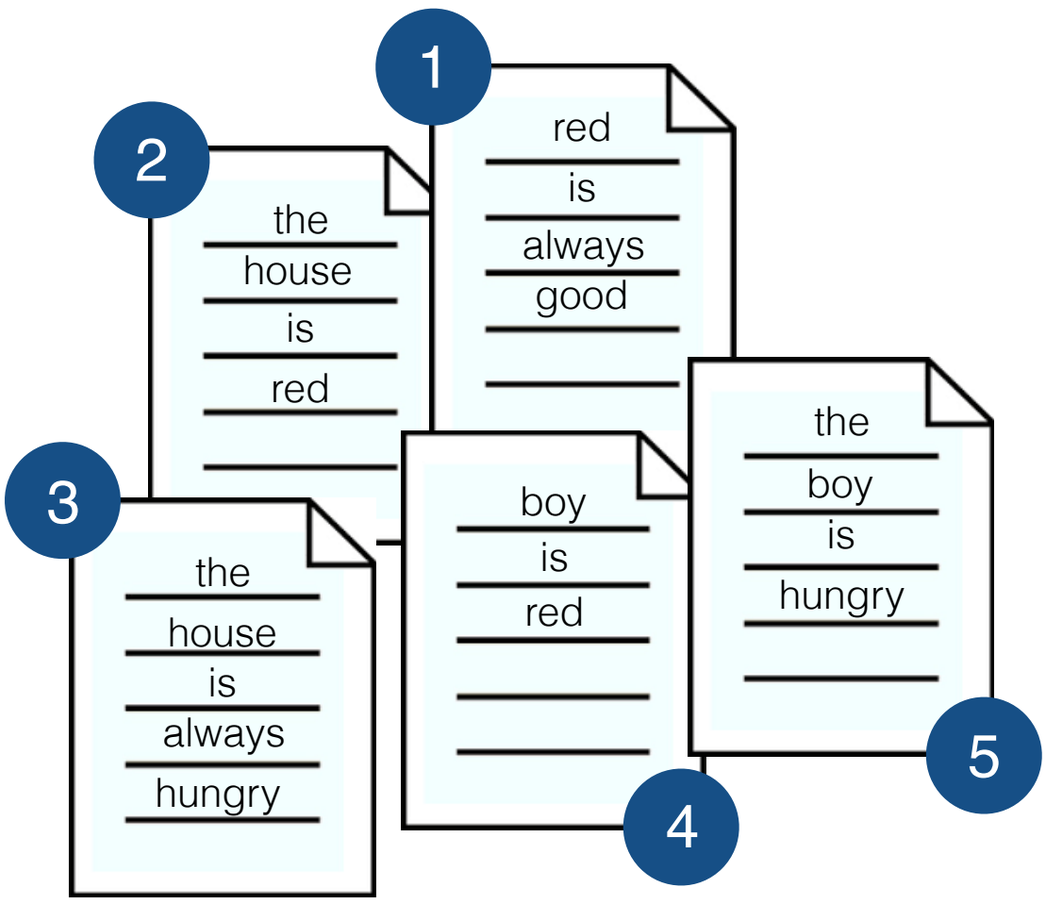


t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8

{always, boy, good, house, hungry, is, red, the}

Inverted indexes

The inverted index is the *de-facto* data structure at the basis of every large-scale retrieval system.



t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8

{always, boy, good, house, hungry, is, red, the}

$L_{t_1}=[1, 3]$
$L_{t_2}=[4, 5]$
$L_{t_3}=[1]$
$L_{t_4}=[2, 3]$
$L_{t_5}=[3, 5]$
$L_{t_6}=[1, 2, 3, 4, 5]$
$L_{t_7}=[1, 2, 4]$
$L_{t_8}=[2, 3, 5]$

Many solutions

Large research corpora describing different **space/time** trade-offs.

- Elias' Gamma and Delta
- Variable-Byte Family
- Binary Interpolative Coding
- Simple Family
- PForDelta
- QMX
- Elias-Fano
- Partitioned Elias-Fano

~1970

2014

Many solutions

Large research corpora describing different **space/time** trade-offs.

- Elias' Gamma and Delta
- Variable-Byte Family
- Binary Interpolative Coding
- Simple Family
- PForDelta
- QMX
- Elias-Fano
- Partitioned Elias-Fano

~1970

2014

Space

Binary Interpolative Coding

~**3X** smaller

Spectrum

Time

Variable-Byte Family

~**4.5X** faster

Key research questions

Space

Binary Interpolative Coding (BIC)

~**3X** smaller

← Spectrum →

Time

Variable-Byte (VByte) Family

~**4.5X** faster

Key research questions

Space

Time

Binary Interpolative Coding (BIC) ← Spectrum → **Variable-Byte (VByte) Family**

~**3X** smaller

~**4.5X** faster

1 Is it possible to design an encoding that is **as small as BIC** and **much faster**?

Key research questions

Space ← Spectrum → **Time**

Binary Interpolative Coding (BIC)

~**3X** smaller

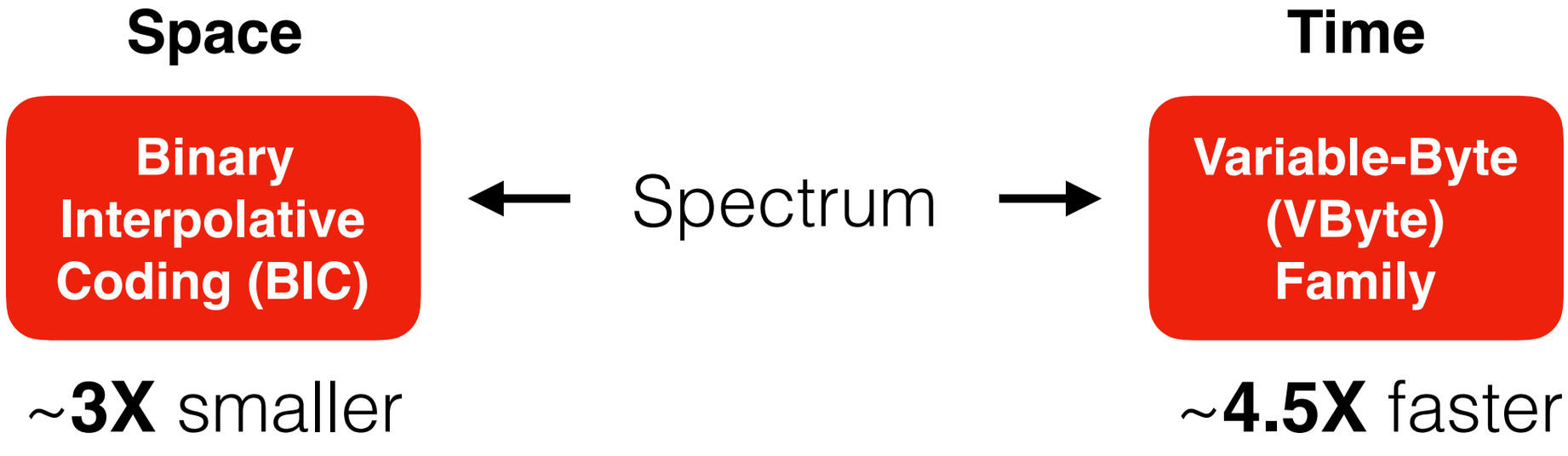
Variable-Byte (VByte) Family

~**4.5X** faster

1 Is it possible to design an encoding that is **as small as BIC** and **much faster**?

2 Is it possible to design an encoding that is **as fast as VByte** and **much smaller**?

Key research questions



1. Is it possible to design an encoding that is **as small as BIC** and **much faster**?

2. Is it possible to design an encoding that is **as fast as VByte** and **much smaller**?

3. What about **both** objectives at the same time?!

Key research questions

Space ← Spectrum → **Time**

Binary Interpolative Coding (BIC)

~**3X** smaller

Variable-Byte (VByte) Family

~**4.5X** faster

Is it possible to design an encoding that is **as small as BIC** and **much faster**?

1

TOIS 2017

Is it possible to design an encoding that is **as fast as VByte** and **much smaller**?

2

TKDE 2019

What about **both** objectives at the same time?!

3

WSDM 2019

1 - Clustered inverted indexes (TOIS 2017)

Every encoder represents each sequence **individually**.

1 - Clustered inverted indexes (TOIS 2017)

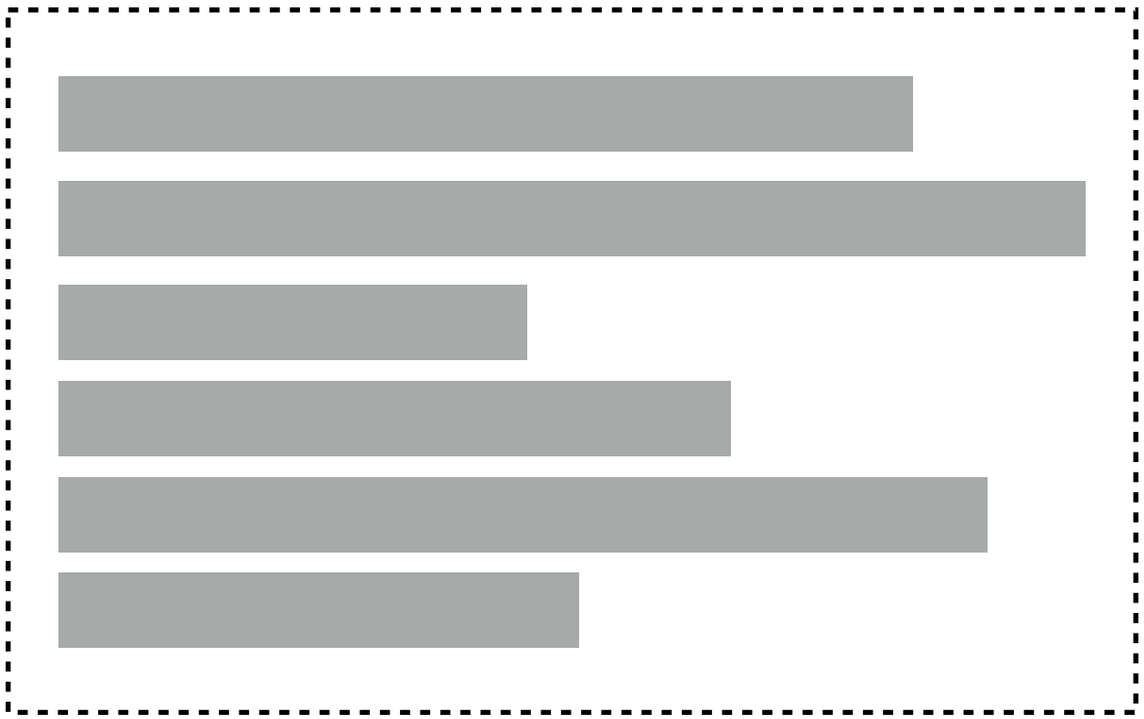
Every encoder represents each sequence **individually**.

Encode **clusters** of (similar) inverted lists.

1 - Clustered inverted indexes (TOIS 2017)

Every encoder represents each sequence **individually**.

Encode **clusters** of (similar) inverted lists.

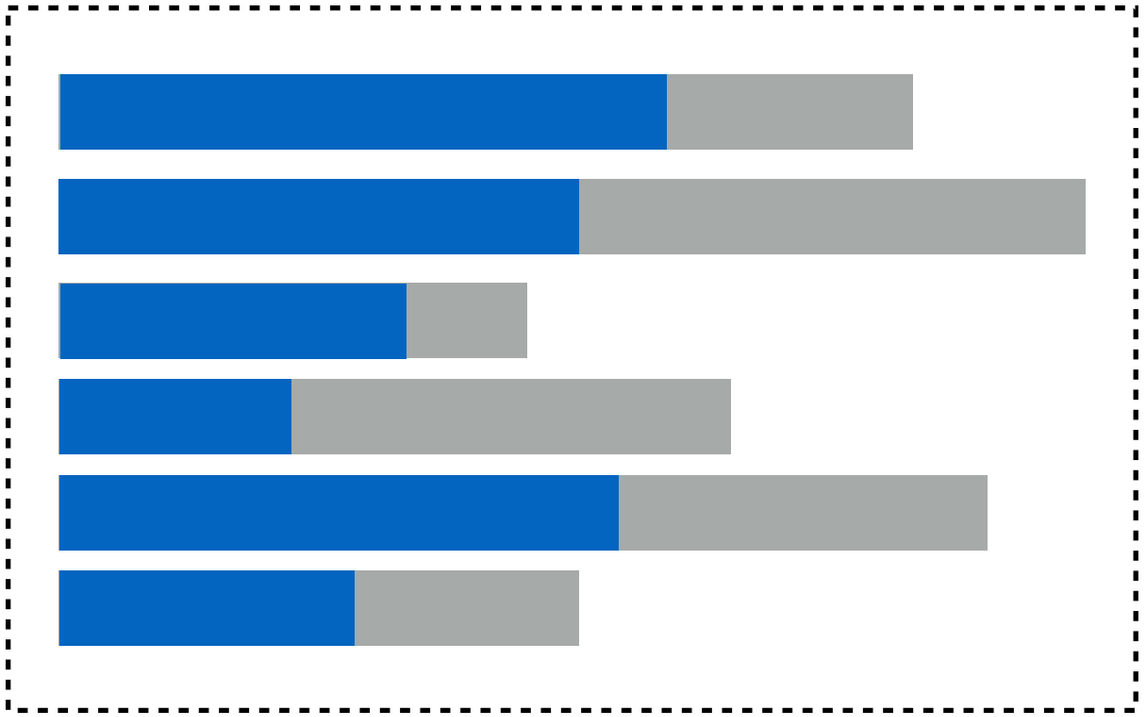


reference list

1 - Clustered inverted indexes (TOIS 2017)

Every encoder represents each sequence **individually**.

Encode **clusters** of (similar) inverted lists.

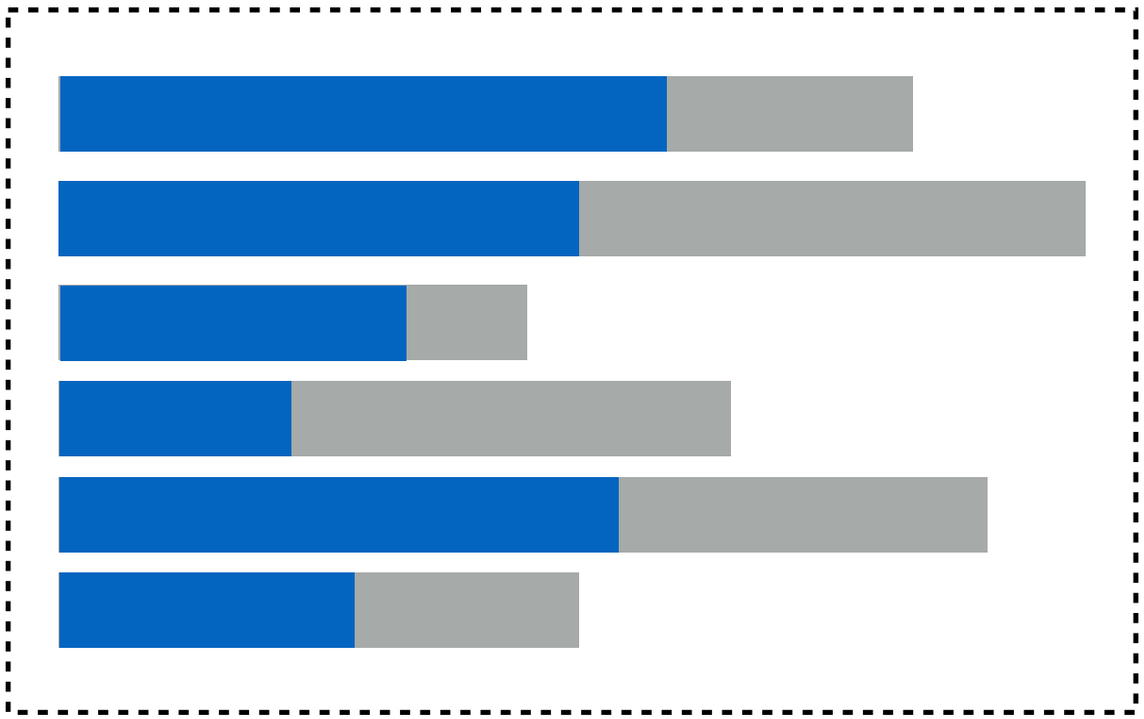


reference list

1 - Clustered inverted indexes (TOIS 2017)

Every encoder represents each sequence **individually**.

Encode **clusters** of (similar) inverted lists.



reference list

Space

Always better than PEF (by up to **11%**) and better than BIC (by up to **6.25%**)

← Spectrum →

Time

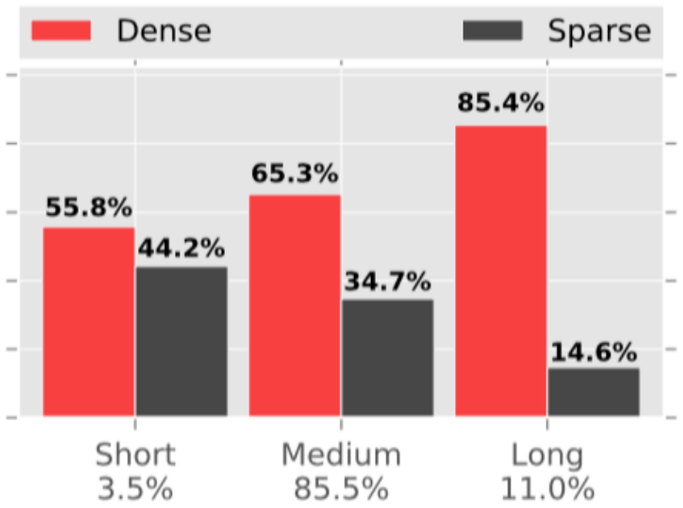
Slightly slower than PEF (~20%) **Much faster** than BIC (2X)

2 - Optimally-partitioned Variable-Byte codes (TKDE 2019)

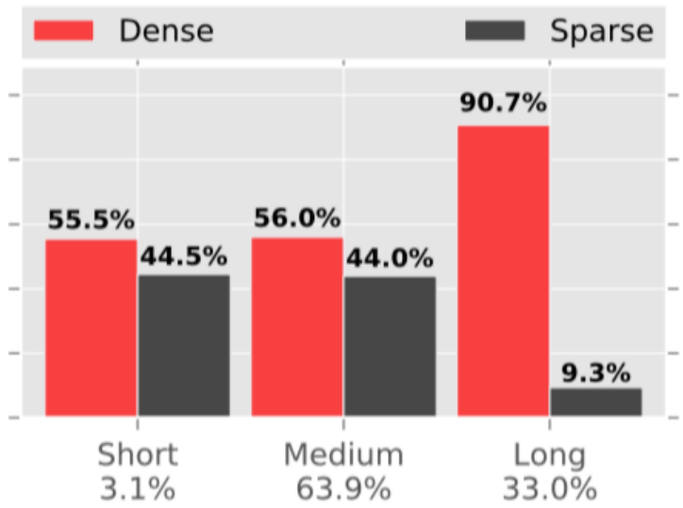
The majority of values are **small** (*very* small indeed).

2 - Optimally-partitioned Variable-Byte codes (TKDE 2019)

The majority of values are **small** (*very* small indeed).



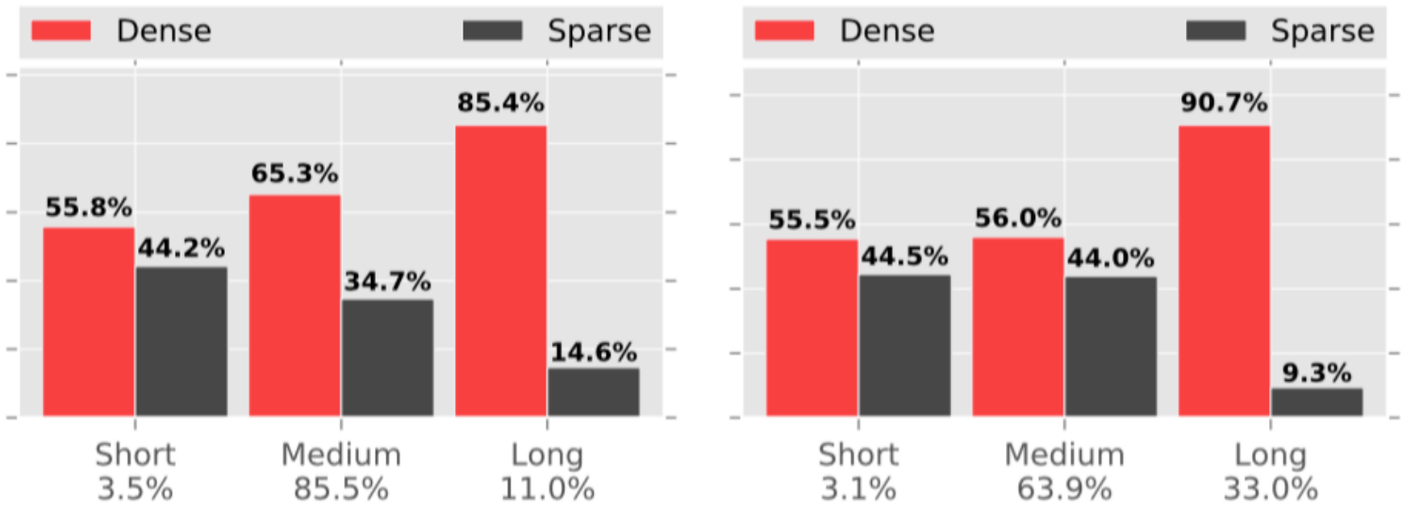
(a) Gov2



(b) ClueWeb09

2 - Optimally-partitioned Variable-Byte codes (TKDE 2019)

The majority of values are **small** (*very* small indeed).



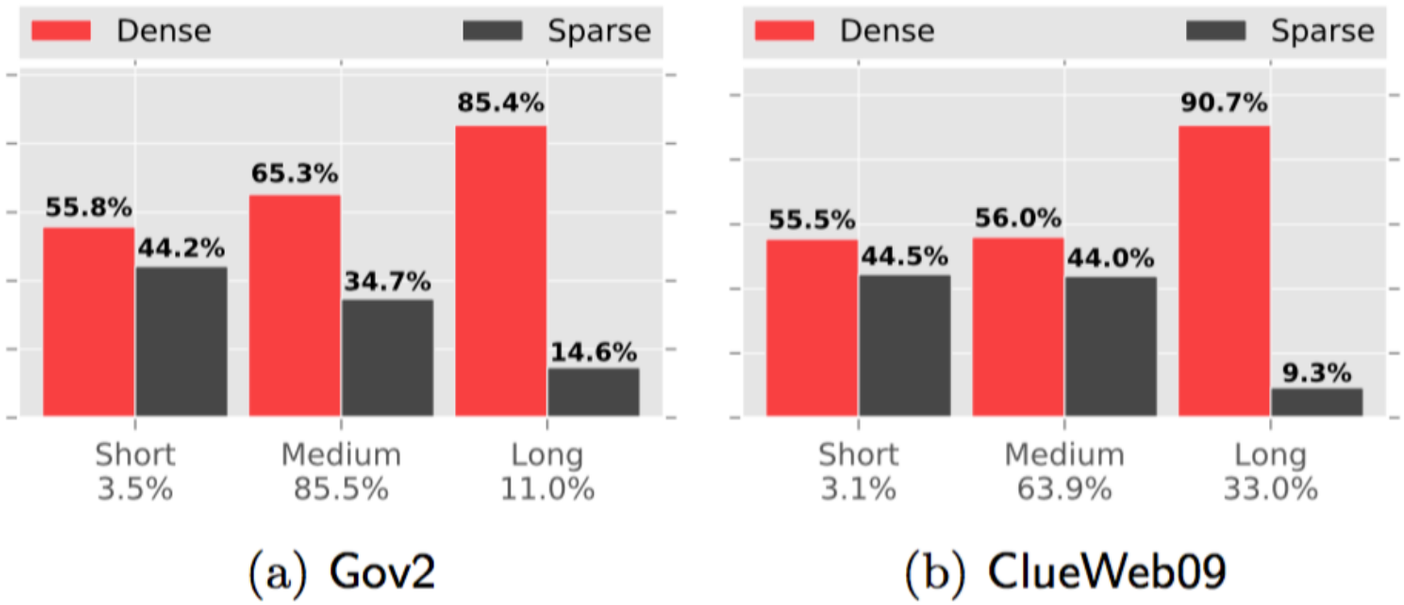
(a) Gov2

(b) ClueWeb09

Encode **dense** regions with unary codes, **sparse** regions with VByte.

2 - Optimally-partitioned Variable-Byte codes (TKDE 2019)

The majority of values are **small** (*very* small indeed).



(a) Gov2 (b) ClueWeb09

Encode **dense** regions with unary codes, **sparse** regions with VByte.

Optimal partitioning in linear time and constant space.

Compression ratio improves by **2X**.

Query processing speed and sequential decoding (almost) **not affected**.

3 - Dictionary-based compression (WSDM 2019)

If we consider subsequences of *d*-gaps in inverted lists,
these are **repetitive** across the whole inverted index.

3 - Dictionary-based compression (WSDM 2019)

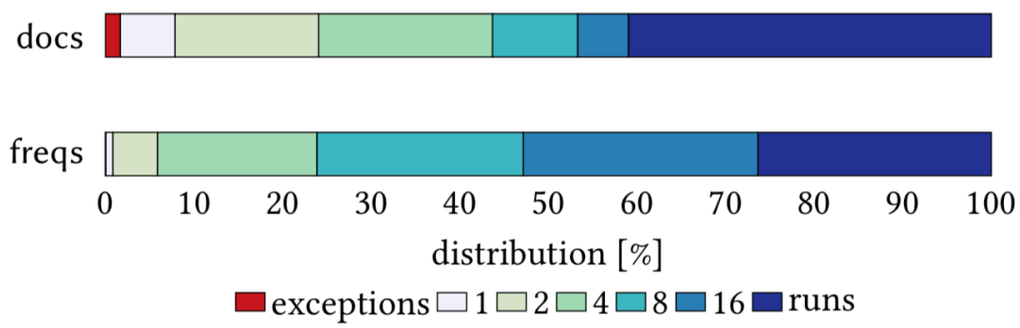
If we consider subsequences of *d*-gaps in inverted lists,
these are **repetitive** across the whole inverted index.

Put the **most frequent patterns**
in a *dictionary* of size k.
Then encode inverted lists as
sequences of $\log_2 k$-bit codewords.

3 - Dictionary-based compression (WSDM 2019)

If we consider subsequences of *d*-gaps in inverted lists,
these are **repetitive** across the whole inverted index.

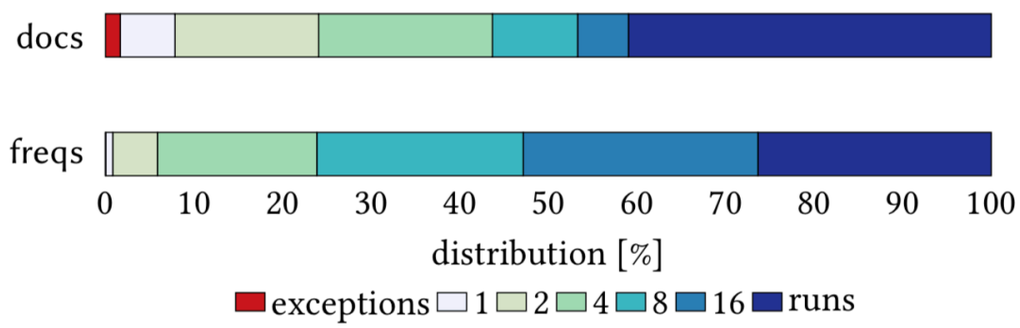
Put the **most frequent patterns**
in a *dictionary* of size *k*.
Then encode inverted lists as
sequences of $\log_2 k$-bit codewords.



3 - Dictionary-based compression (WSDM 2019)

If we consider subsequences of *d*-gaps in inverted lists, these are **repetitive** across the whole inverted index.

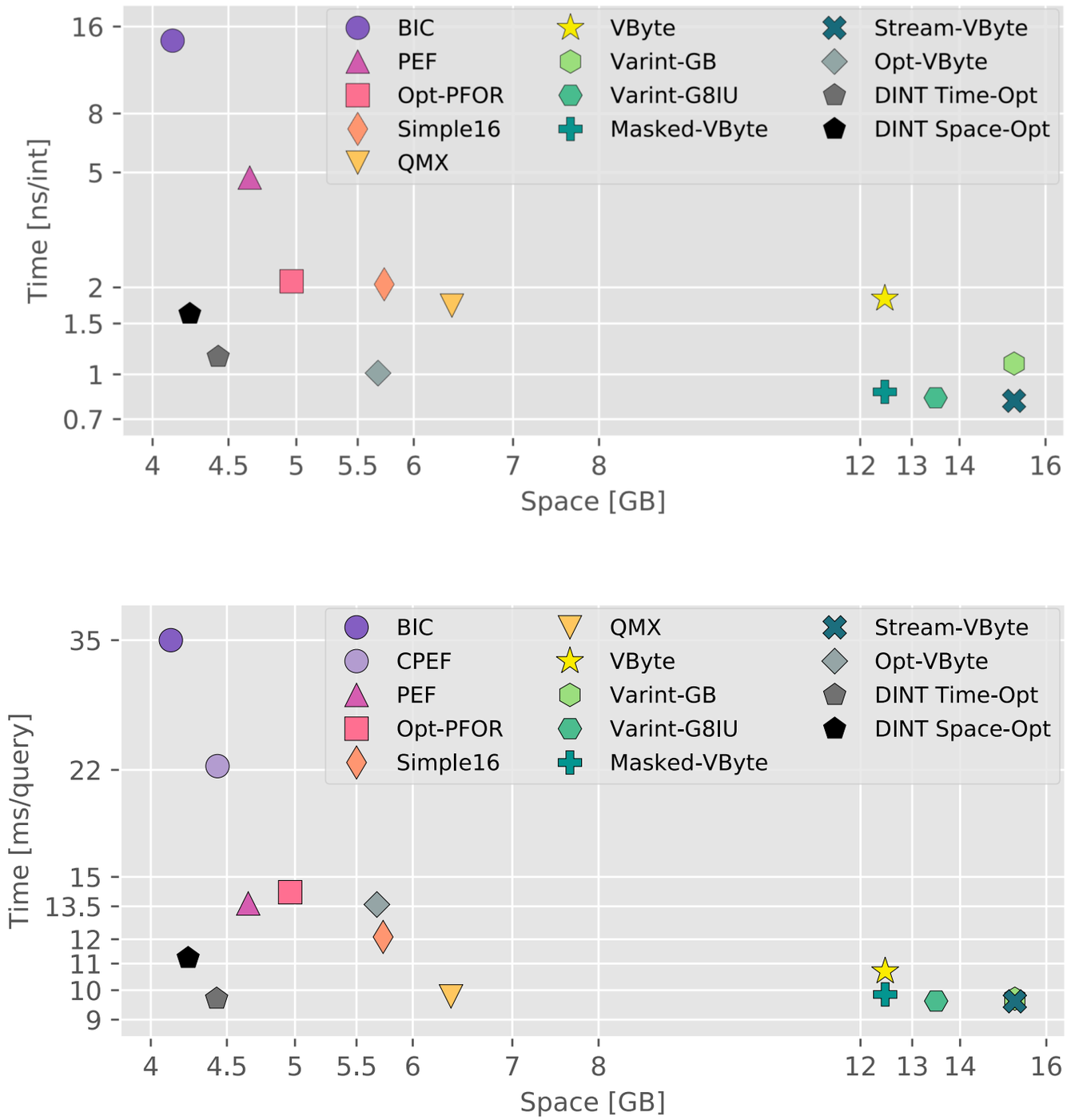
Put the **most frequent patterns** in a *dictionary* of size *k*. Then encode inverted lists as sequences of $\log_2 k$-bit codewords.



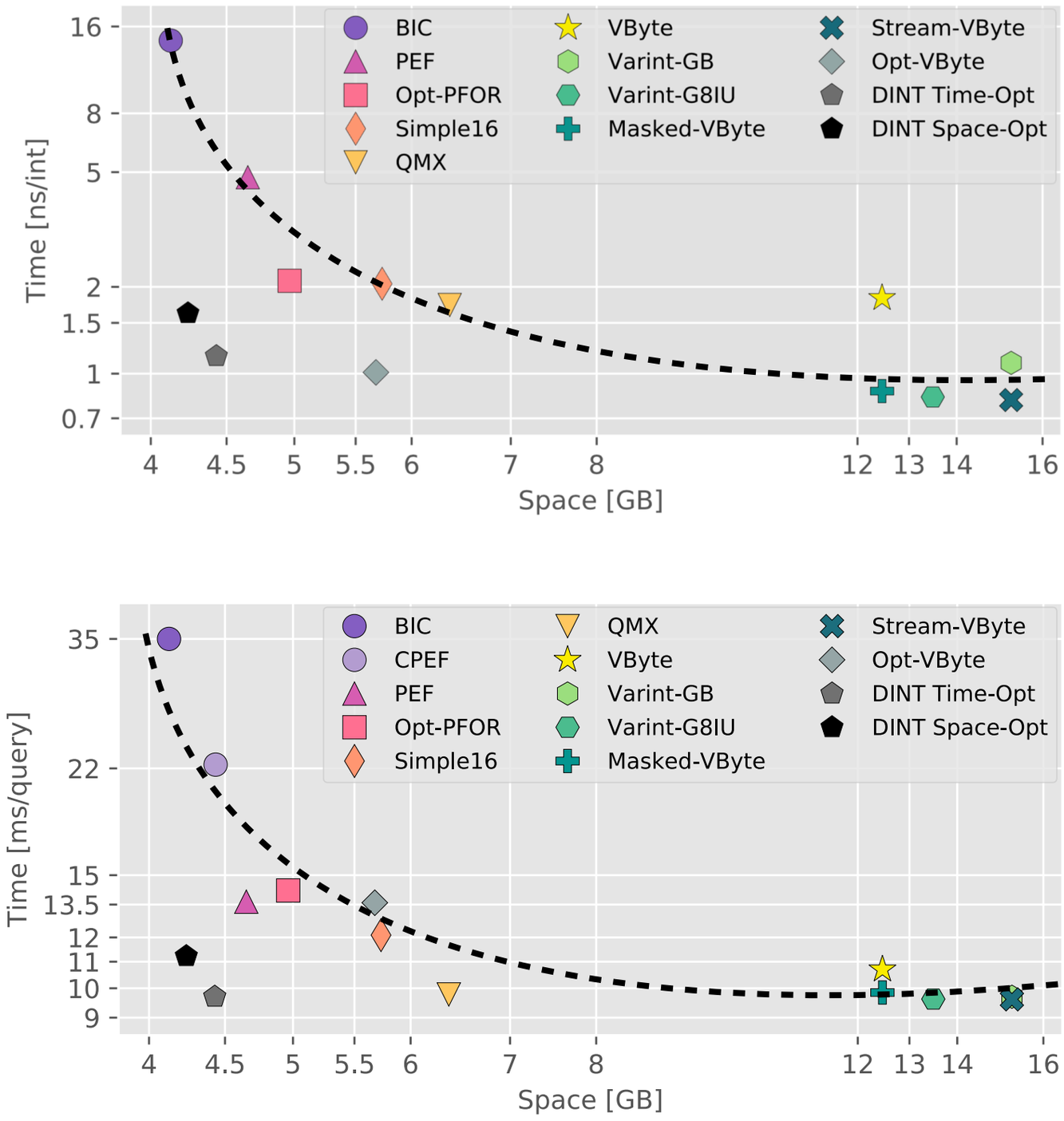
Close to the most space-efficient representation (~**7%** away from BIC).

Almost **as fast as** the fastest SIMD-ized decoders.

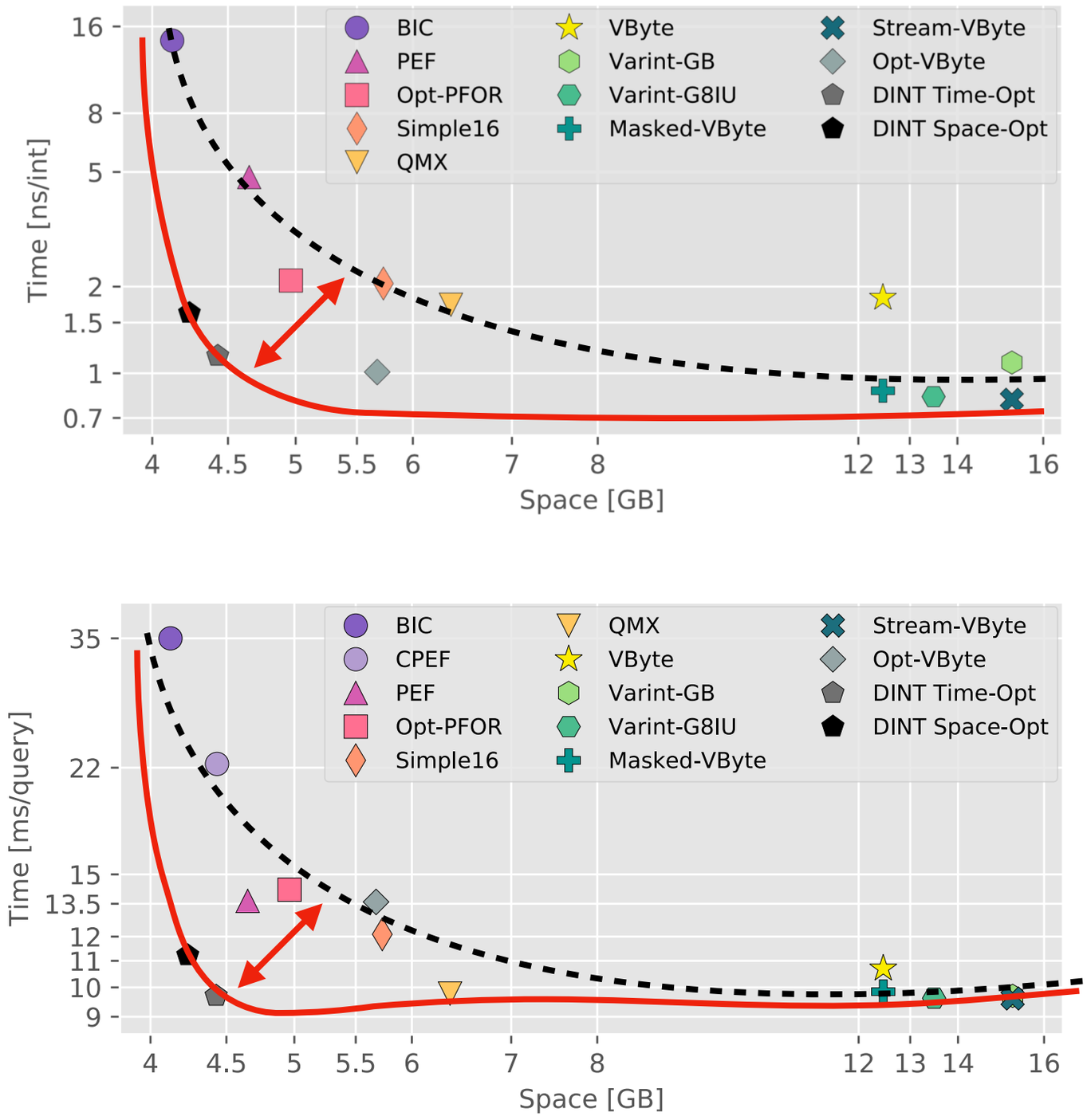
The bigger picture



The bigger picture



The bigger picture



Problem 2

Consider a large text.

Problem 2

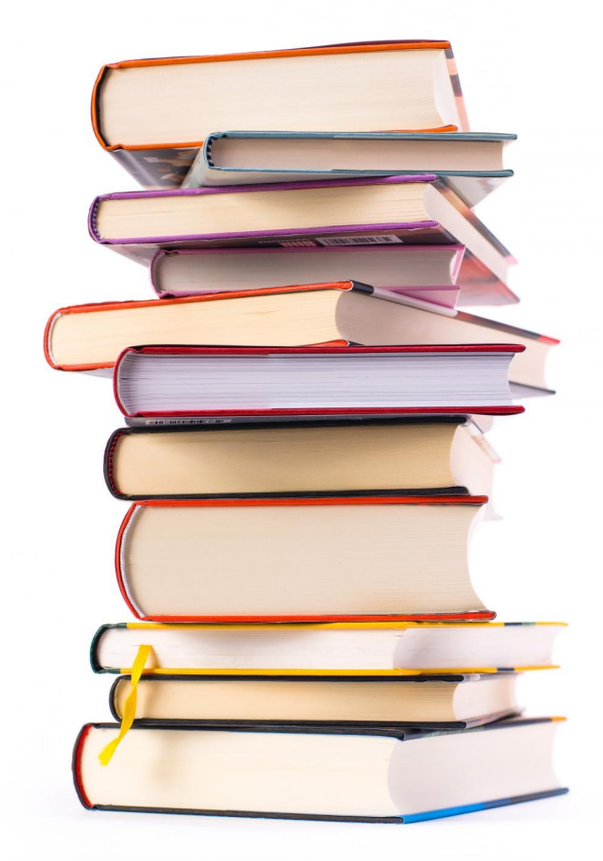
Consider a large text.

How to represent all its substrings of size $1 \leq k \leq N$ words for fixed N (e.g., N = 5), using **as few as possible** bits?

Fast **access** to individual N-grams?

How to **estimate** the probability of occurrence of the patterns under a given probability model?

Indexing



Google Books

~6% of the books ever published

N	number of N-grams
1	24,359,473
2	667,284,771
3	7,397,041,901
4	1,644,807,896
5	1,415,355,596

More than 11 billions of N-grams!

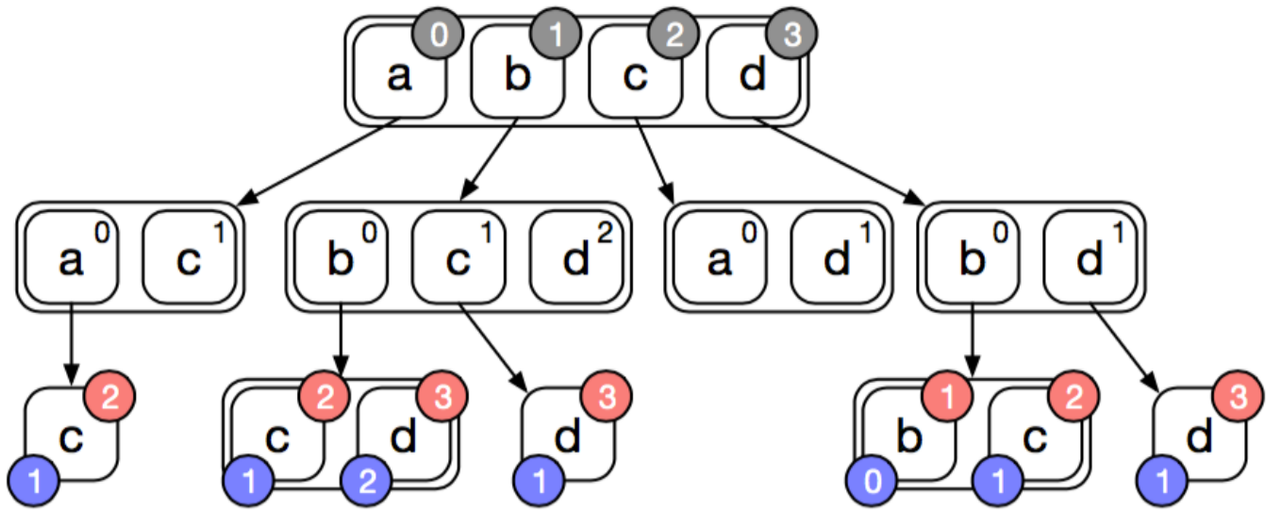
Context-based remapped tries (SIGIR 2017)

The number of words following a given context is **small**.

Context-based remapped tries (SIGIR 2017)

The number of words following a given context is **small**.

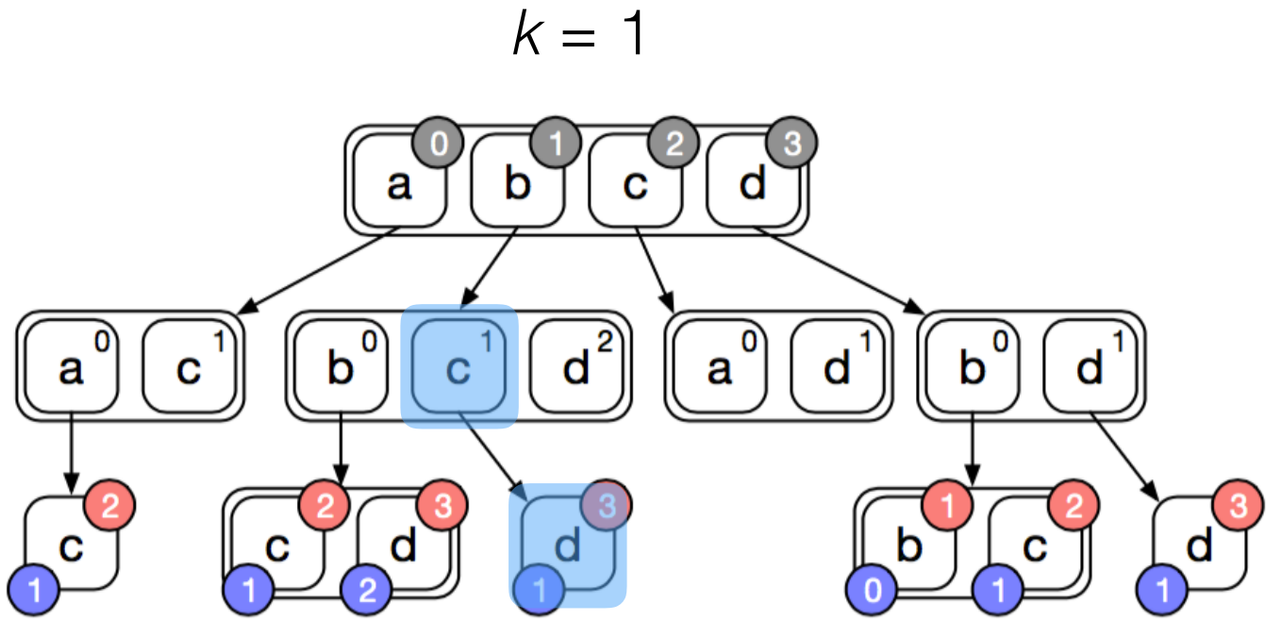
$k = 1$



Map a word ID to the **position** it takes within its *sibling* IDs (the IDs following a context of fixed length *k*).

Context-based remapped tries (SIGIR 2017)

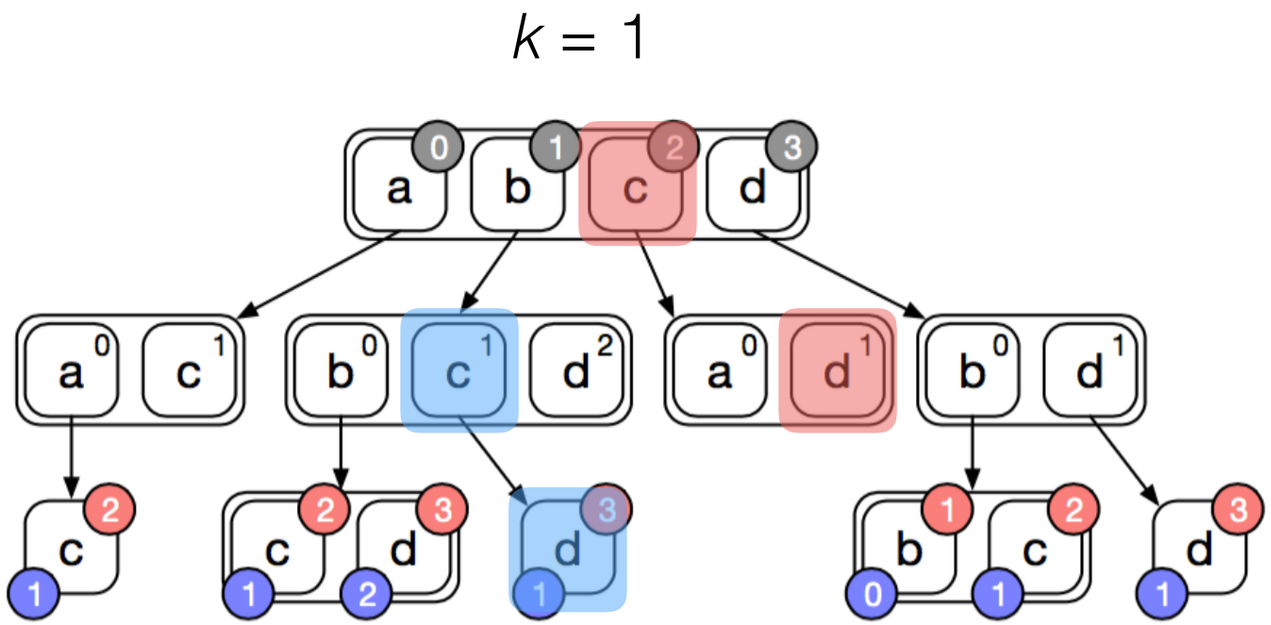
The number of words following a given context is **small**.



Map a word ID to the **position** it takes within its *sibling* IDs (the IDs following a context of fixed length *k*).

Context-based remapped tries (SIGIR 2017)

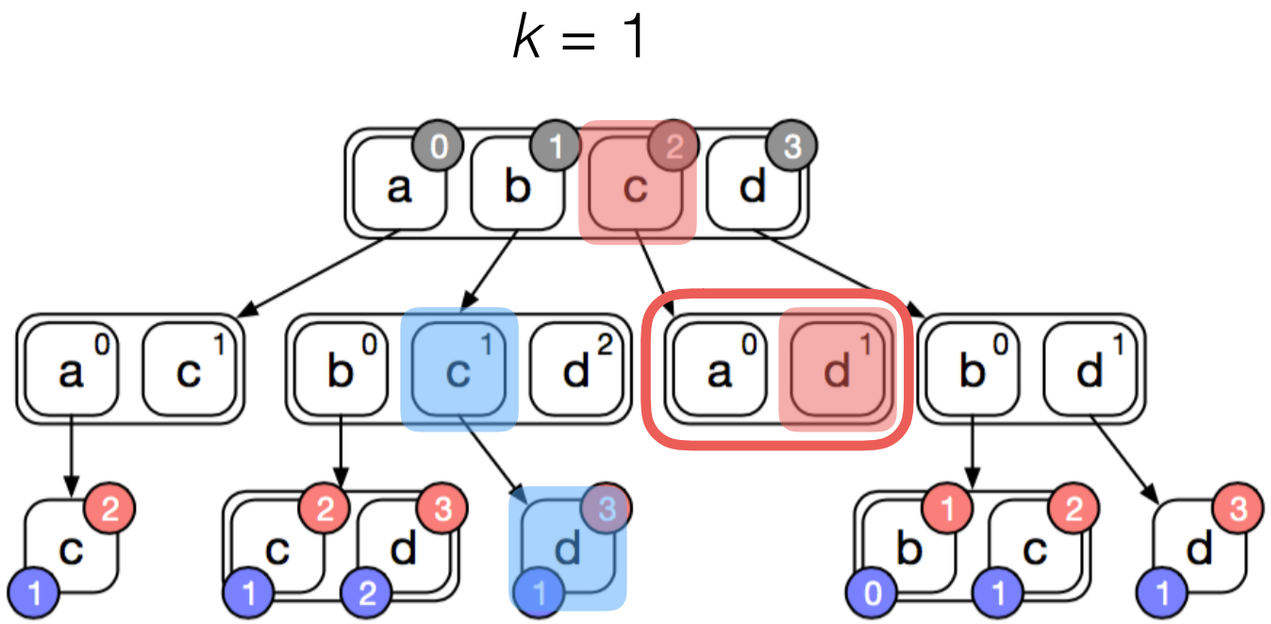
The number of words following a given context is **small**.



Map a word ID to the **position** it takes within its *sibling* IDs (the IDs following a context of fixed length *k*).

Context-based remapped tries (SIGIR 2017)

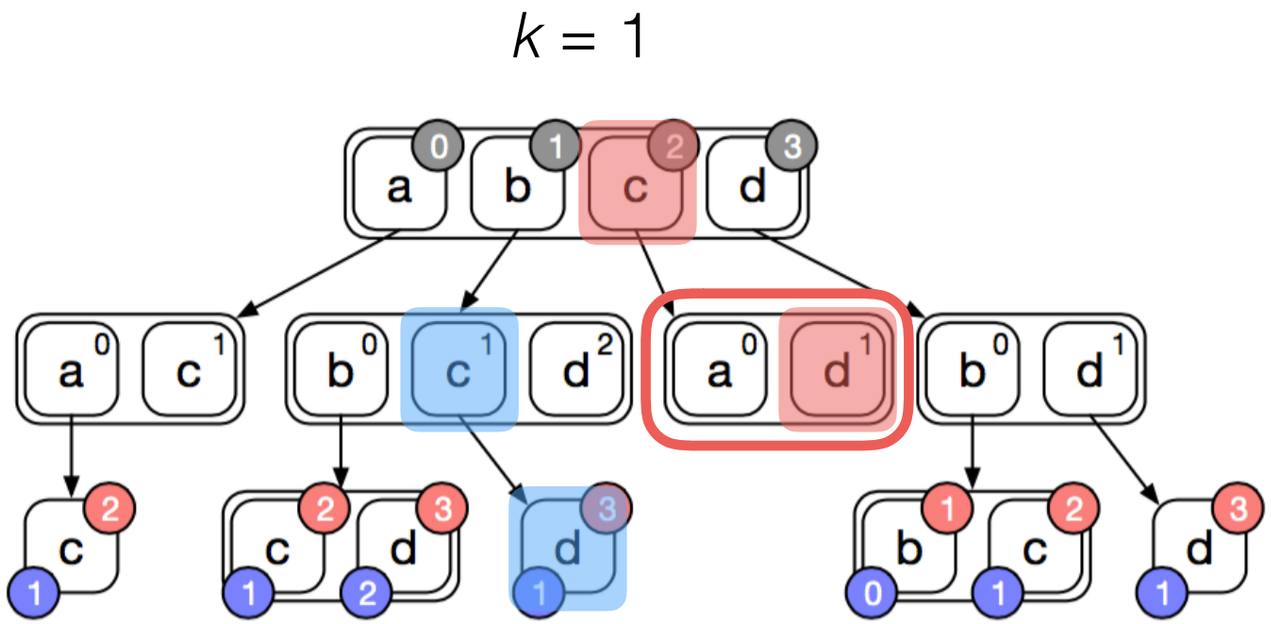
The number of words following a given context is **small**.



Map a word ID to the **position** it takes within its *sibling* IDs (the IDs following a context of fixed length *k*).

Context-based remapped tries (SIGIR 2017)

The number of words following a given context is **small**.

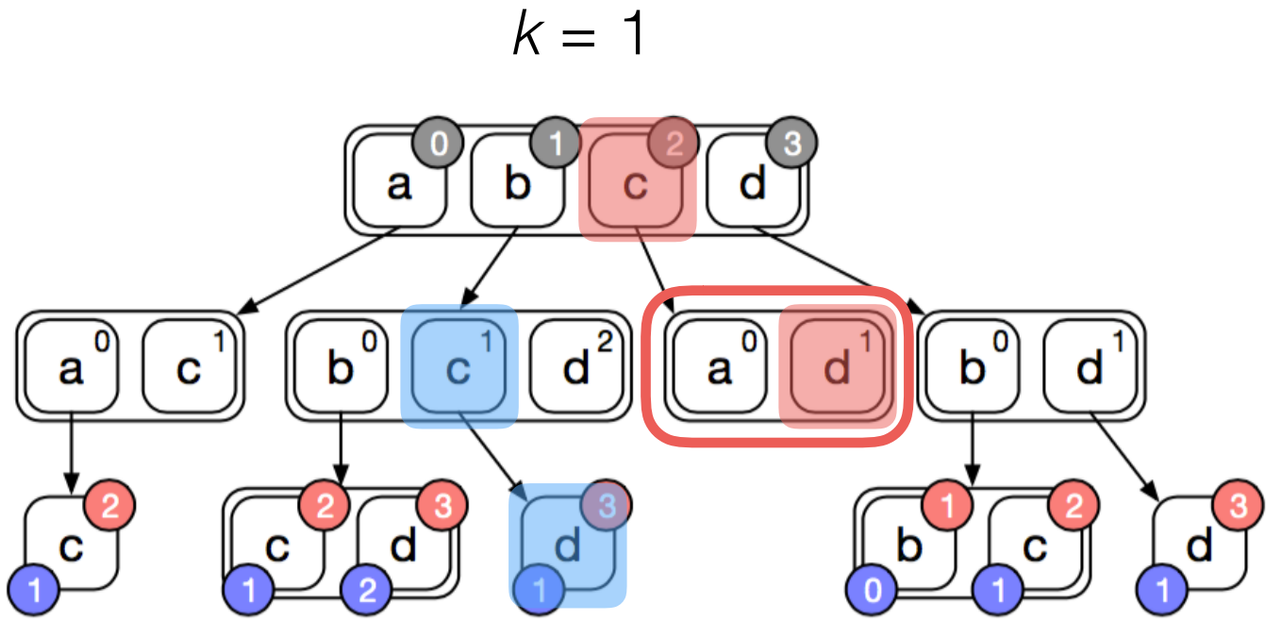


Map a word ID to the **position** it takes within its *sibling* IDs (the IDs following a context of fixed length *k*).

The (Elias-Fano) context-based remapped trie is **as fast as** the fastest competitor, but up to **65% smaller**.

Context-based remapped tries (SIGIR 2017)

The number of words following a given context is **small**.



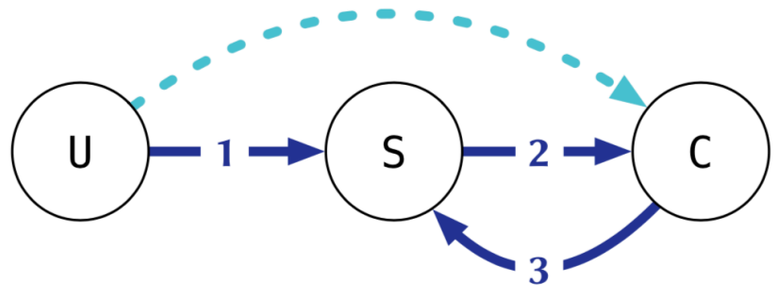
Map a word ID to the **position** it takes within its *sibling* IDs (the IDs following a context of fixed length *k*).

The (Elias-Fano) context-based remapped trie is **as fast as** the fastest competitor, but up to **65% smaller**.

The (Elias-Fano) context-based remapped trie is **even smaller** than the most space-efficient competitors, that are lossy and with false-positives allowed, and up to **5X faster**.

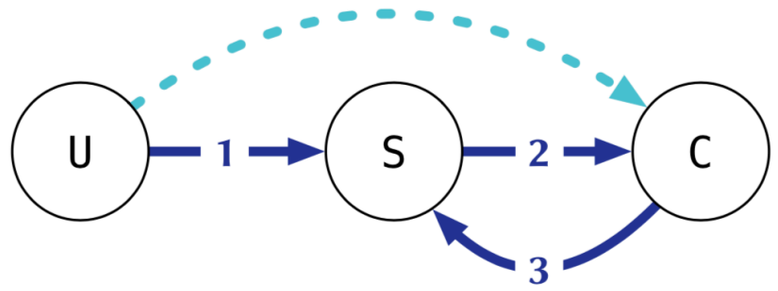
Fast estimation in external memory (TOIS 2019)

To compute the modified Kneser-Ney probabilities of the N-grams, the fastest algorithm in the literature uses **3 sorting steps** in external memory.



Fast estimation in external memory (TOIS 2019)

To compute the modified Kneser-Ney probabilities of the N-grams, the fastest algorithm in the literature uses **3 sorting steps** in external memory.



Computing the distinct left extensions.

1	2	3	4	5	6	7	8	9	10	11	12
A	**A**	**A**	**A**	**B**	**B**	**C**	**C**	**X**	**X**	**X**	**X**
A	B	B	X	A	C	A	A	C	X	X	X
B	A	C	X	X	A	A	B	A	C	X	X
A	X	A	X	X	A	B	C	B	A	C	X
X	X	A	X	X	B	A	A	C	B	A	C

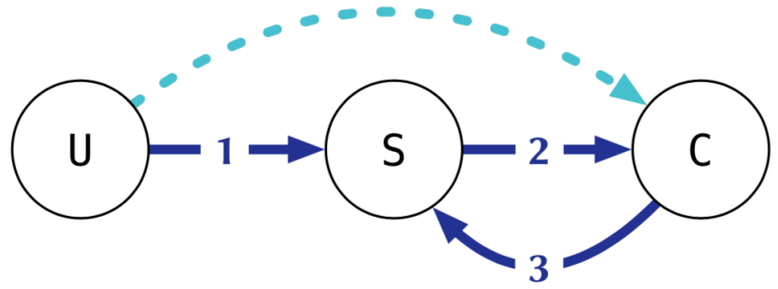
Suffix order

7	1	8	5	2	3	6	9	10	11	12	4
C	A	C	B	A	A	B	X	X	X	X	A
A	**A**	**A**	**A**	**B**	**B**	**C**	**C**	**X**	**X**	**X**	**X**
A	B	B	X	A	C	A	A	C	X	X	X
B	A	C	X	X	A	A	B	A	C	X	X
A	X	A	X	X	A	B	C	B	A	C	X

Context order

Fast estimation in external memory (TOIS 2019)

To compute the modified Kneser-Ney probabilities of the N-grams, the fastest algorithm in the literature uses **3 sorting steps** in external memory.



Computing the distinct left extensions.

1	2	3	4	5	6	7	8	9	10	11	12
A	**A**	**A**	**A**	**B**	**B**	**C**	**C**	**X**	**X**	**X**	**X**
A	B	B	X	A	C	A	A	C	X	X	X
B	A	C	X	X	A	A	B	A	C	X	X
A	X	A	X	X	A	B	C	B	A	C	X
X	X	A	X	X	B	A	A	C	B	A	C

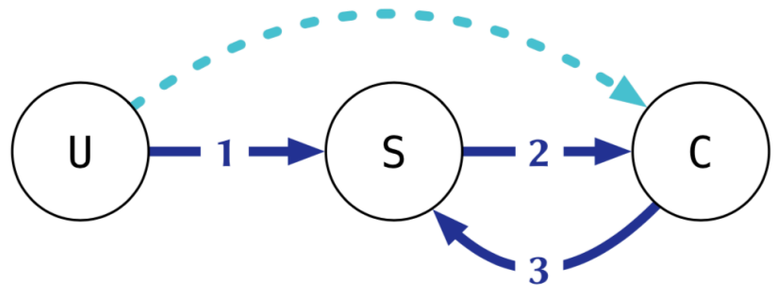
Suffix order

7	1	8	5	2	3	6	9	10	11	12	4
C	A	C	B	A	A	B	X	X	X	X	A
A	**A**	**A**	**A**	**B**	**B**	**C**	**C**	**X**	**X**	**X**	**X**
A	B	B	X	A	C	A	A	C	X	X	X
B	A	C	X	X	A	A	B	A	C	X	X
A	X	A	X	X	A	B	C	B	A	C	X

Context order

Fast estimation in external memory (TOIS 2019)

To compute the modified Kneser-Ney probabilities of the N-grams, the fastest algorithm in the literature uses **3 sorting steps** in external memory.



Computing the distinct left extensions.

1	2	3	4	5	6	7	8	9	10	11	12
A	**A**	**A**	**A**	**B**	**B**	**C**	**C**	**X**	**X**	**X**	**X**
A	B	B	X	A	C	A	A	C	X	X	X
B	A	C	X	X	A	A	B	A	C	X	X
A	X	A	X	X	A	B	C	B	A	C	X
X	X	A	X	X	B	A	A	C	B	A	C

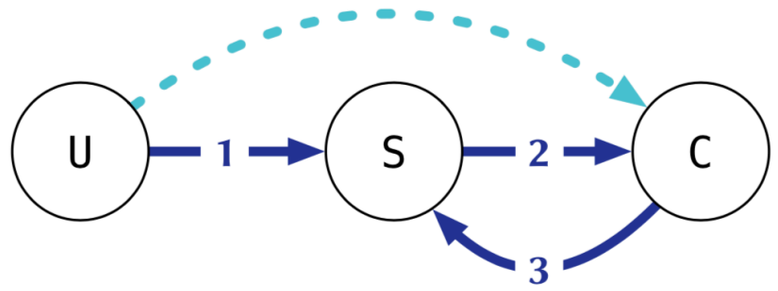
Suffix order

7	1	8	5	2	3	6	9	10	11	12	4
C	A	C	B	A	A	B	X	X	X	X	A
A	**A**	**A**	**A**	**B**	**B**	**C**	**C**	**X**	**X**	**X**	**X**
A	B	B	X	A	C	A	A	C	X	X	X
B	A	C	X	X	A	A	B	A	C	X	X
A	X	A	X	X	A	B	C	B	A	C	X

Context order

Fast estimation in external memory (TOIS 2019)

To compute the modified Kneser-Ney probabilities of the N-grams, the fastest algorithm in the literature uses **3 sorting steps** in external memory.



Computing the distinct left extensions.

1	2	3	4	5	6	7	8	9	10	11	12
A	**A**	**A**	**A**	**B**	**B**	**C**	**C**	**X**	**X**	**X**	**X**
A	B	B	X	A	C	A	A	C	X	X	X
B	A	C	X	X	A	A	B	A	C	X	X
A	X	A	X	X	A	B	C	B	A	C	X
X	X	A	X	X	B	A	A	C	B	A	C

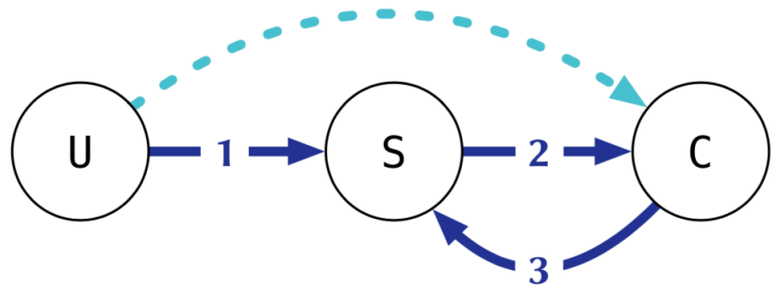
Suffix order

7	1	8	5	2	3	6	9	10	11	12	4
C	A	C	B	A	A	B	X	X	X	X	A
A	**A**	**A**	**A**	**B**	**B**	**C**	**C**	**X**	**X**	**X**	**X**
A	B	B	X	A	C	A	A	C	X	X	X
B	A	C	X	X	A	A	B	A	C	X	X
A	X	A	X	X	A	B	C	B	A	C	X

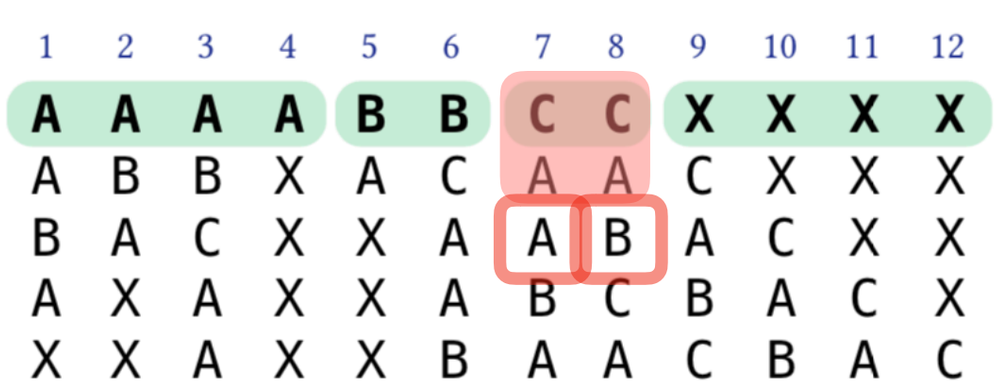
Context order

Fast estimation in external memory (TOIS 2019)

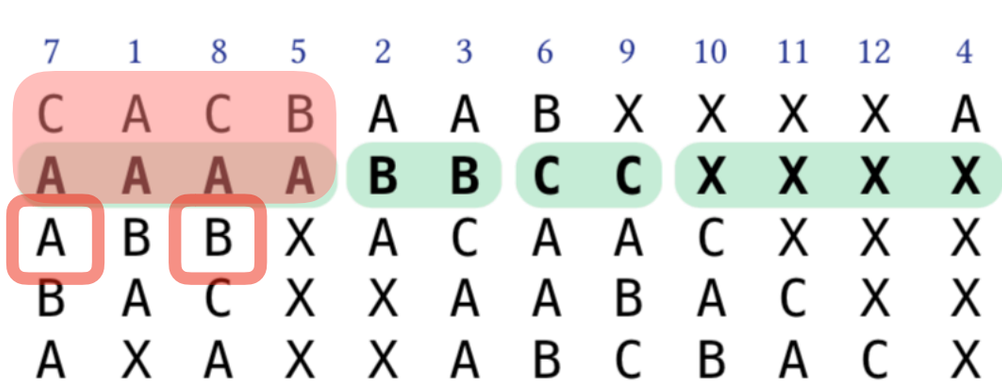
To compute the modified Kneser-Ney probabilities of the N-grams, the fastest algorithm in the literature uses **3 sorting steps** in external memory.



Computing the distinct left extensions.



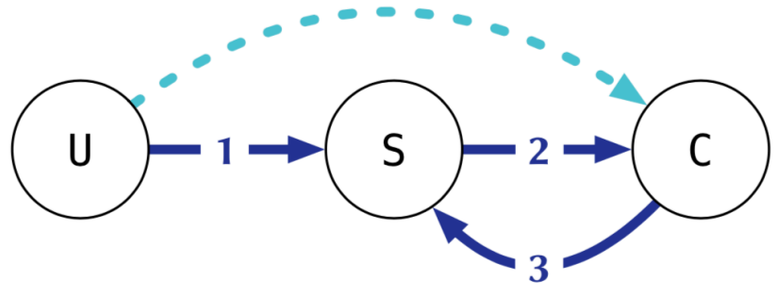
Suffix order



Context order

Fast estimation in external memory (TOIS 2019)

To compute the modified Kneser-Ney probabilities of the N-grams, the fastest algorithm in the literature uses **3 sorting steps** in external memory.



Computing the distinct left extensions.

1	2	3	4	5	6	7	8	9	10	11	12
A	**A**	**A**	**A**	**B**	**B**	**C**	**C**	**X**	**X**	**X**	**X**
A	B	B	X	A	C	A	A	C	X	X	X
B	A	C	X	X	A	A	B	A	C	X	X
A	X	A	X	X	A	B	C	B	A	C	X
X	X	A	X	X	B	A	A	C	B	A	C

Suffix order

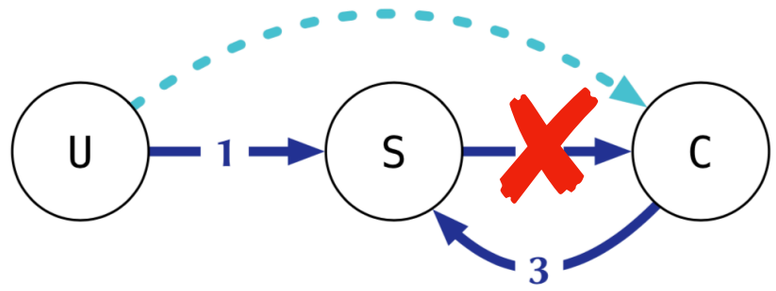
7	1	8	5	2	3	6	9	10	11	12	4
C	A	C	B	A	A	B	X	X	X	X	A
A	**A**	**A**	**A**	**B**	**B**	**C**	**C**	**X**	**X**	**X**	**X**
A	B	B	X	A	C	A	A	C	X	X	X
B	A	C	X	X	A	A	B	A	C	X	X
A	X	A	X	X	A	B	C	B	A	C	X

Context order

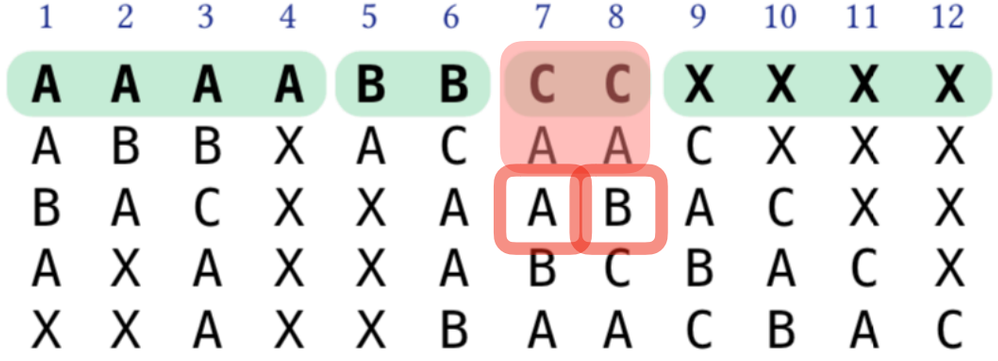
Using a scan of the block and O(|V|) space.

Fast estimation in external memory (TOIS 2019)

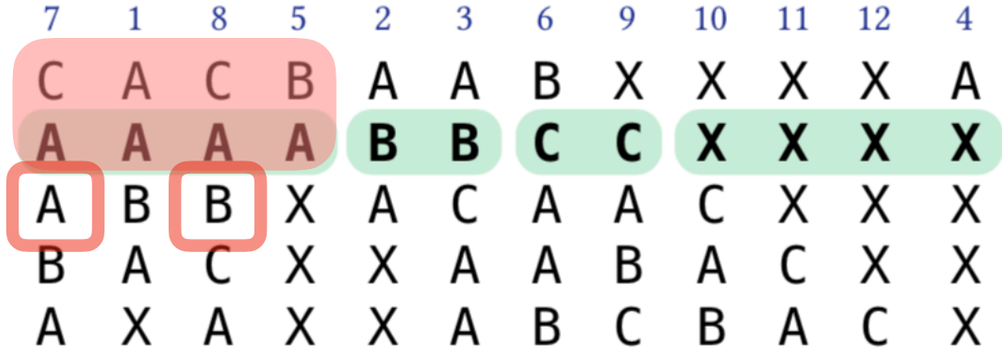
To compute the modified Kneser-Ney probabilities of the N-grams, the fastest algorithm in the literature uses **3 sorting steps** in external memory.



Computing the distinct left extensions.



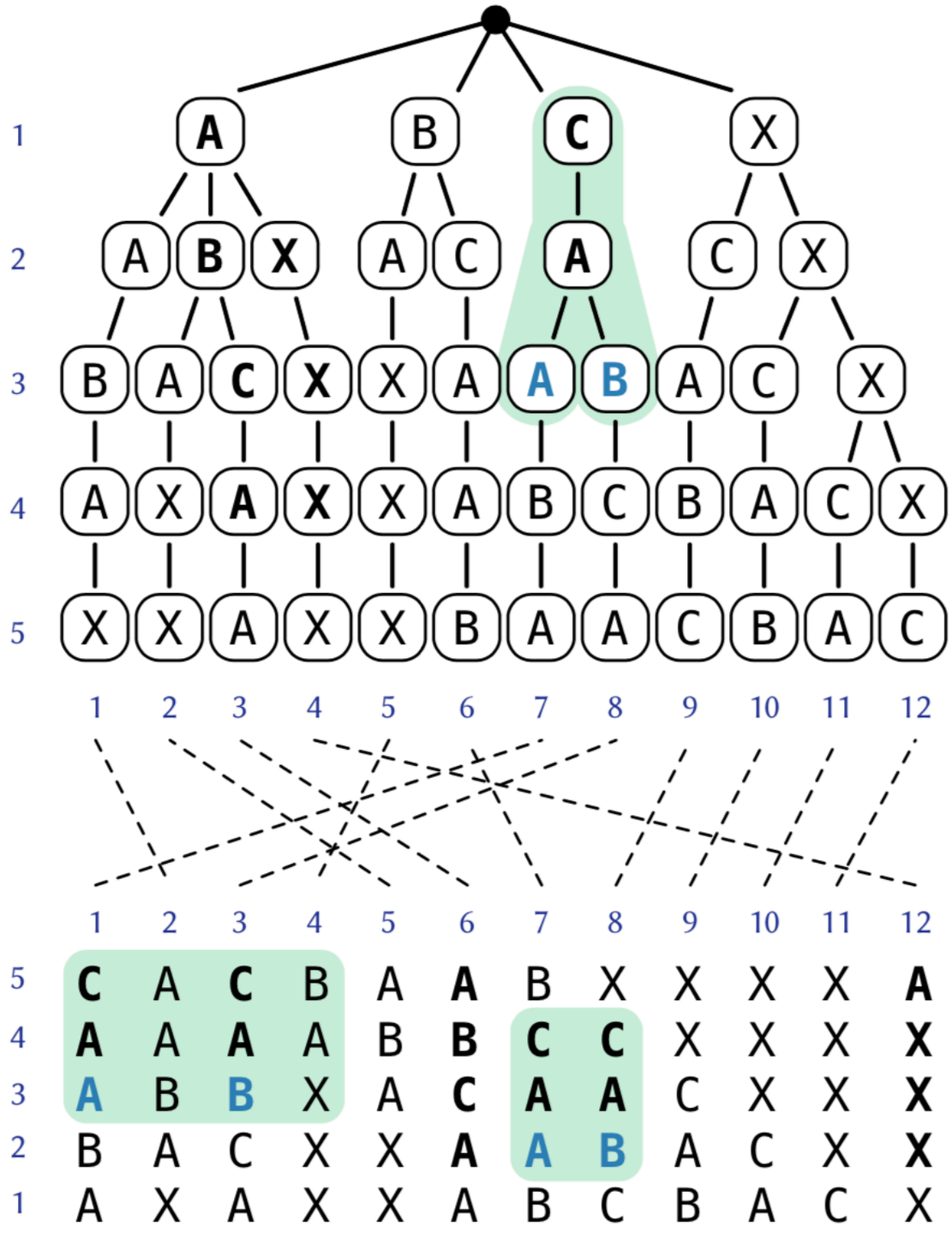
Suffix order

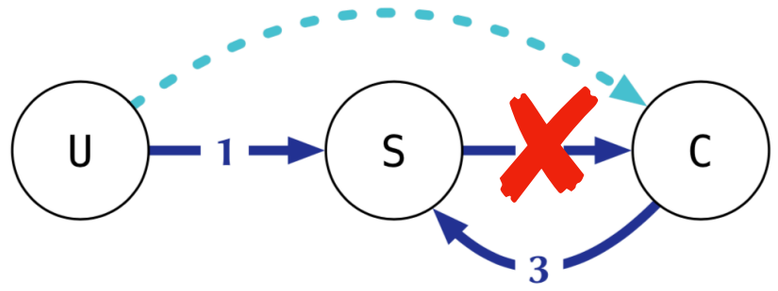


Context order

Using a scan of the block and O(|V|) space.

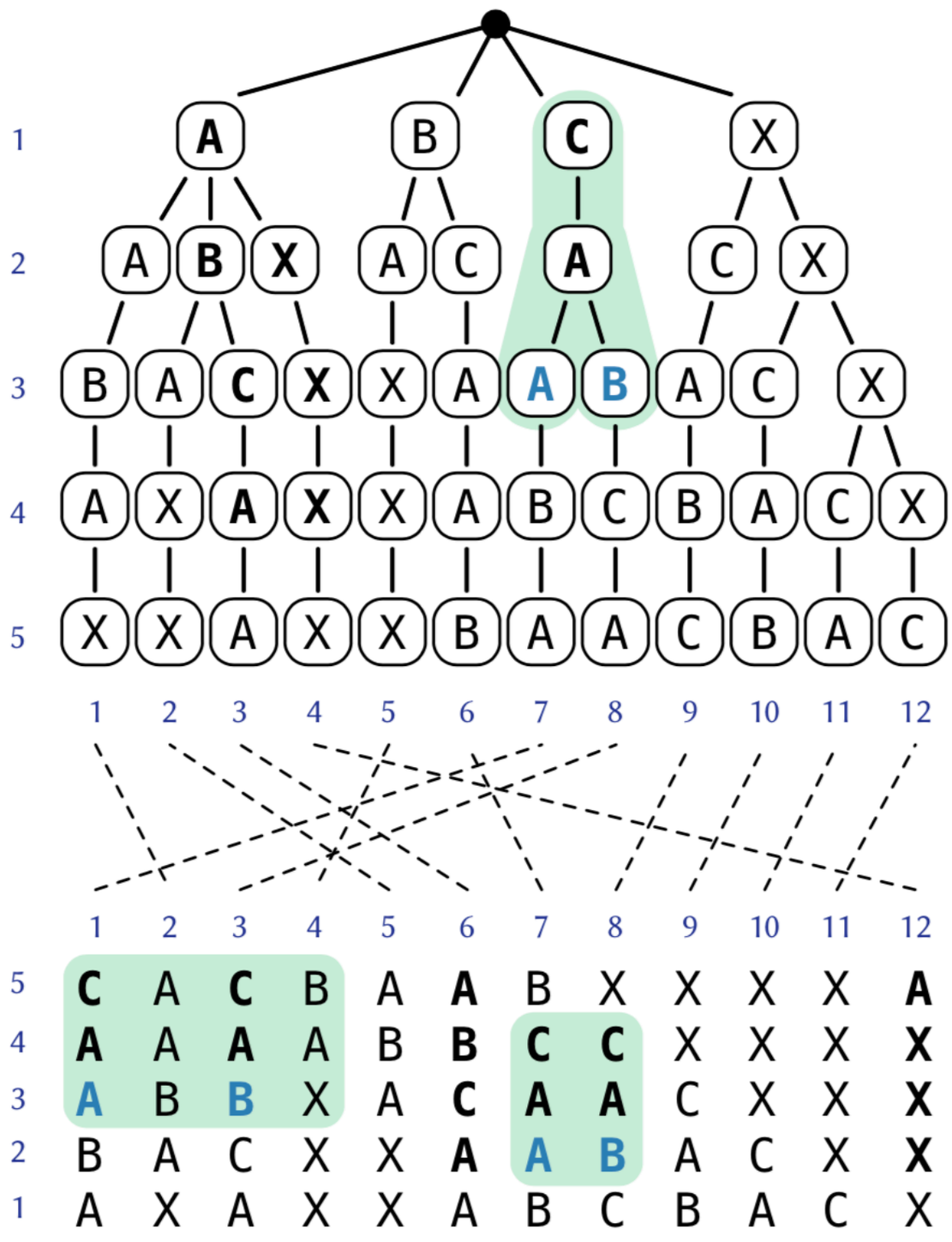
Fast estimation in external memory (TOIS 2019)

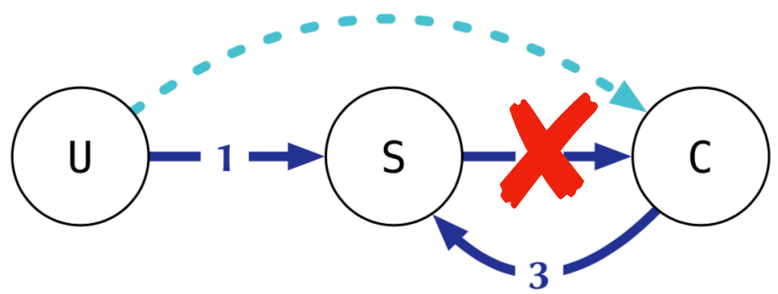




Rebuilding the last level of the trie.

Fast estimation in external memory (TOIS 2019)

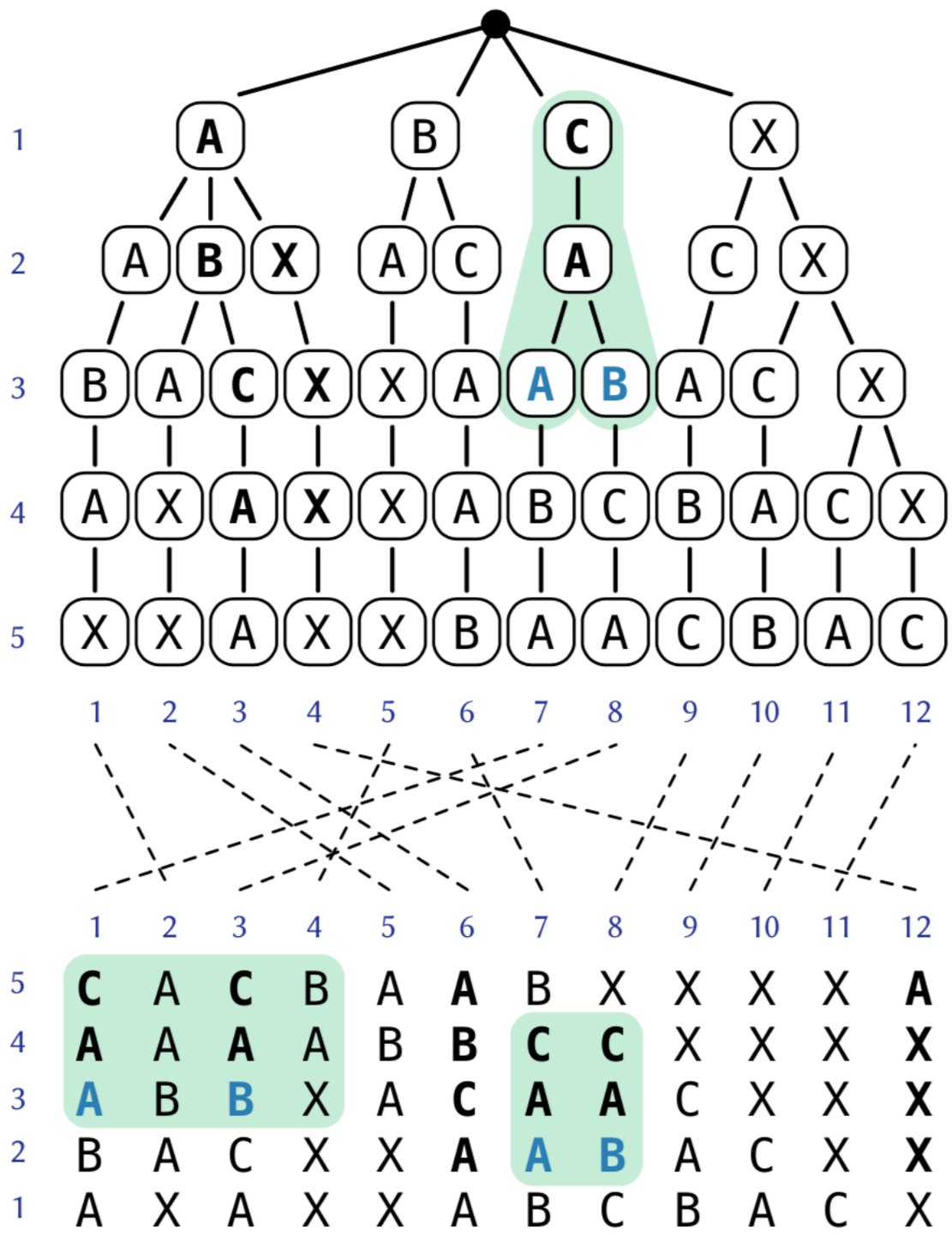


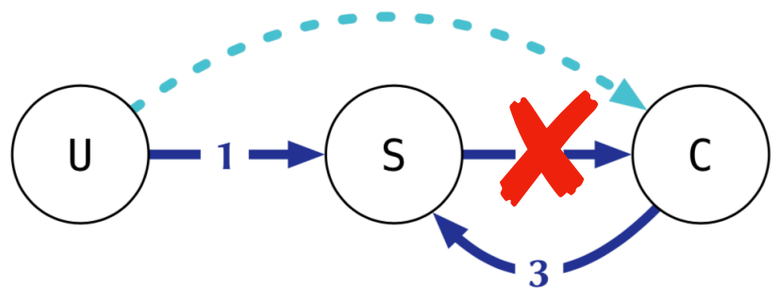


Rebuilding the last level of the trie.

A	4
B	2
C	2
X	4

Fast estimation in external memory (TOIS 2019)





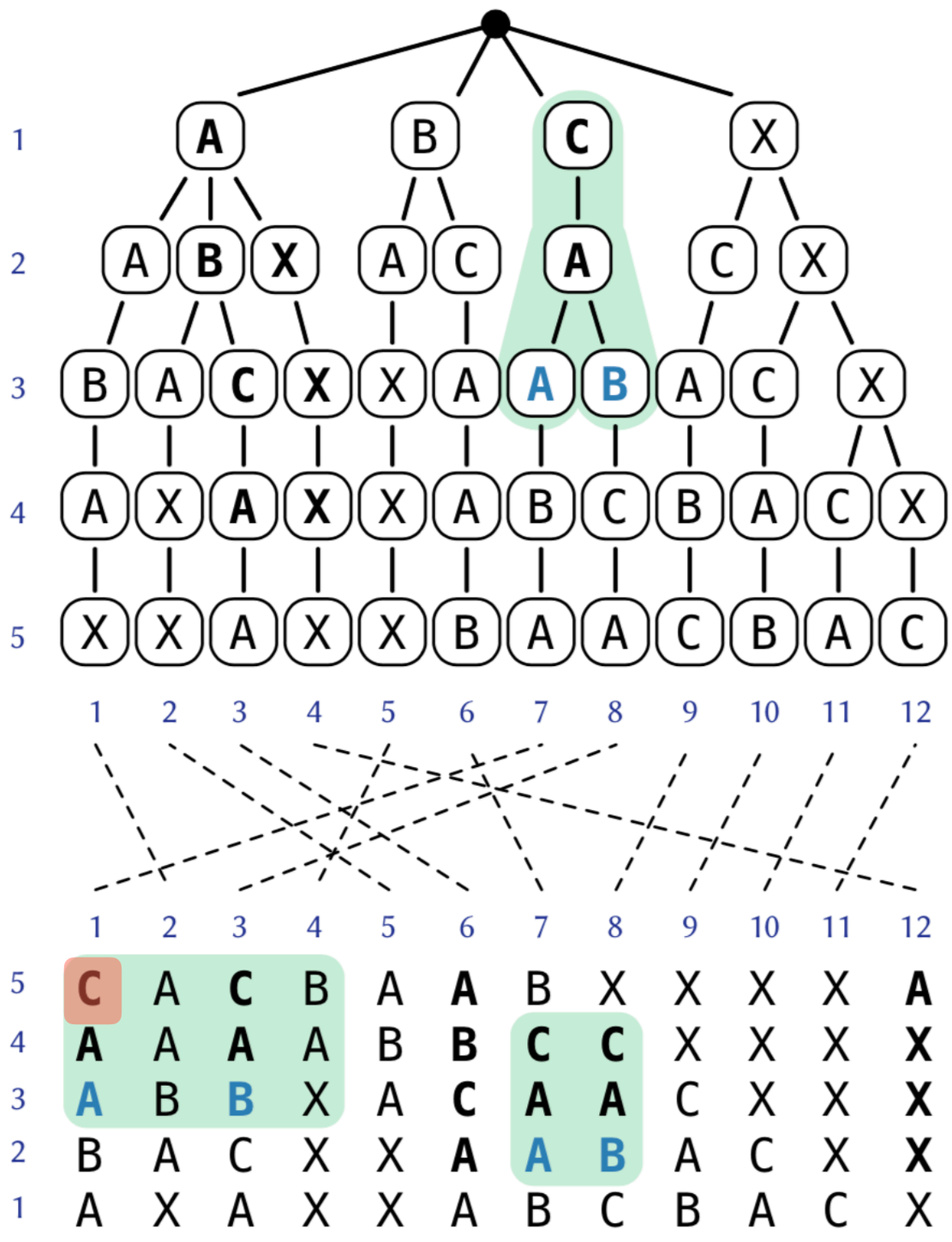
Rebuilding the last level of the trie.

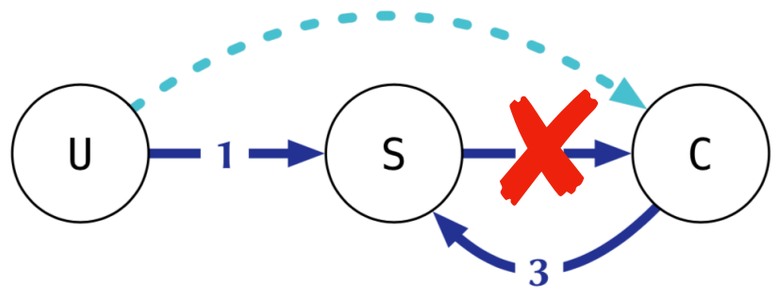
A	4
B	2
C	2
X	4

→

A	1
B	5
C	7
X	9

Fast estimation in external memory (TOIS 2019)





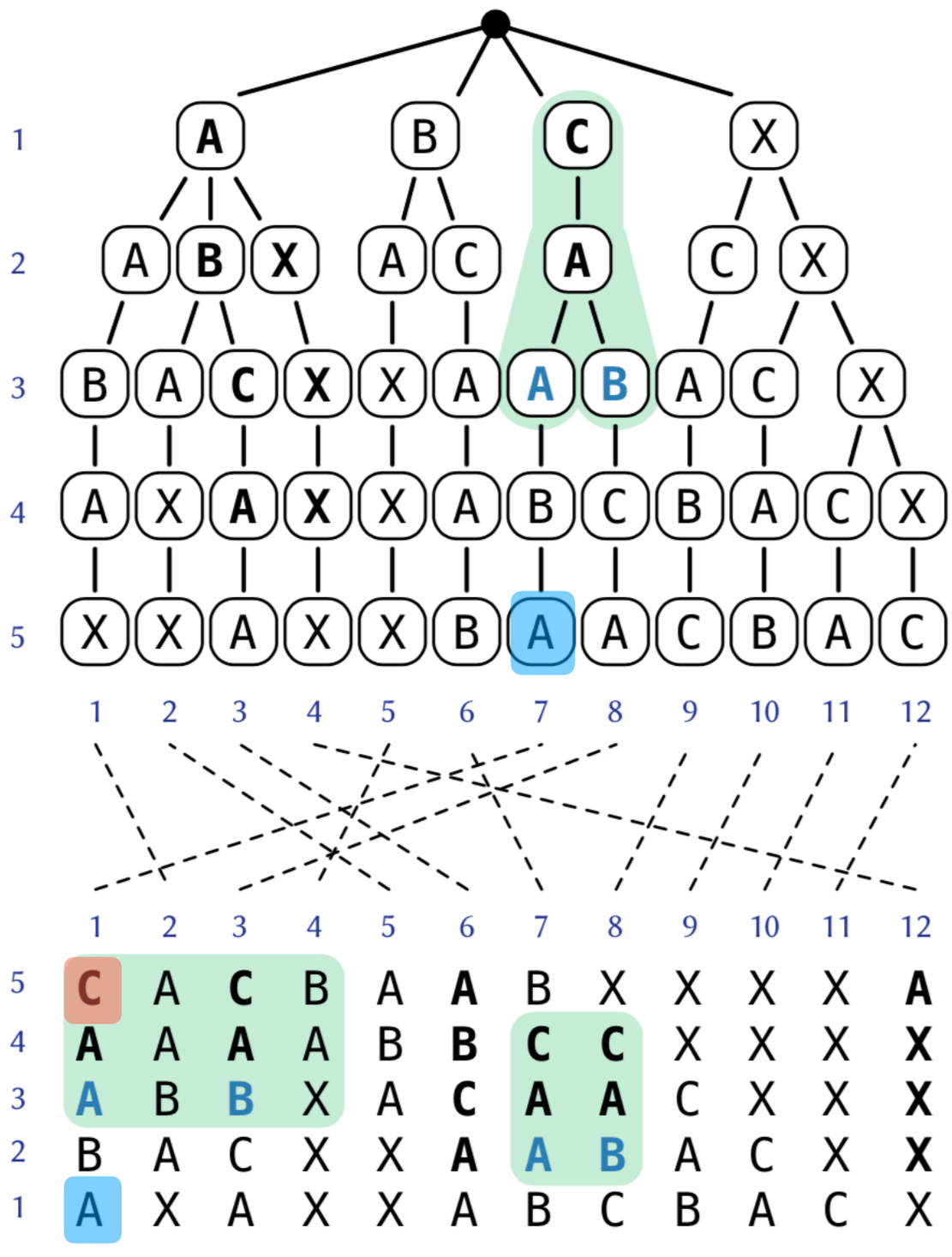
Rebuilding the last level of the trie.

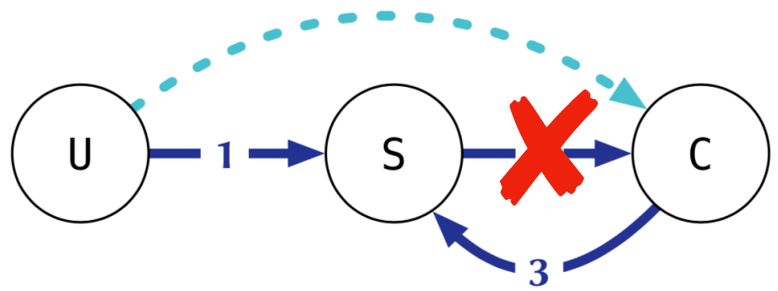
A	4
B	2
C	2
X	4

→

A	1
B	5
C	7
X	9

Fast estimation in external memory (TOIS 2019)





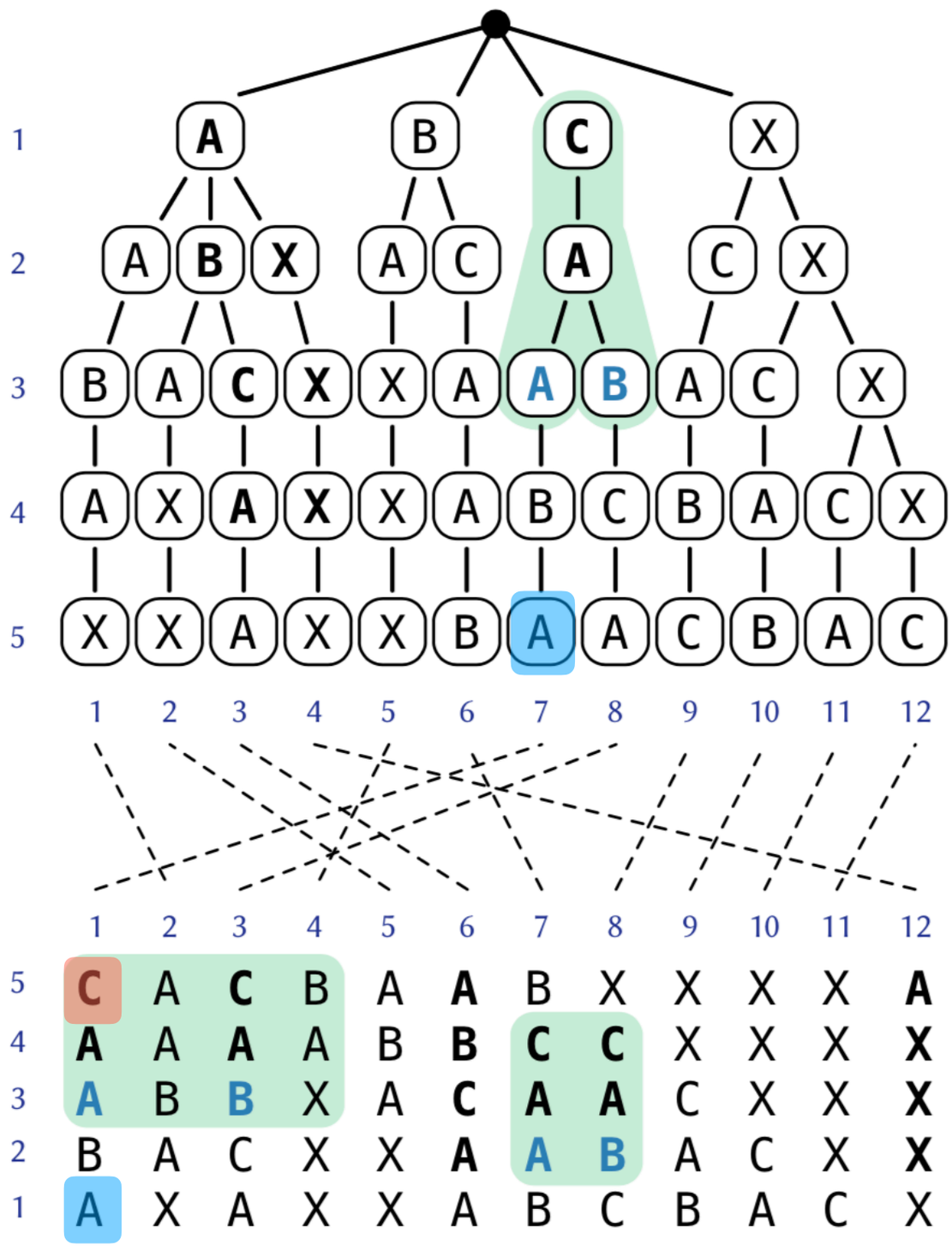
Rebuilding the last level of the trie.

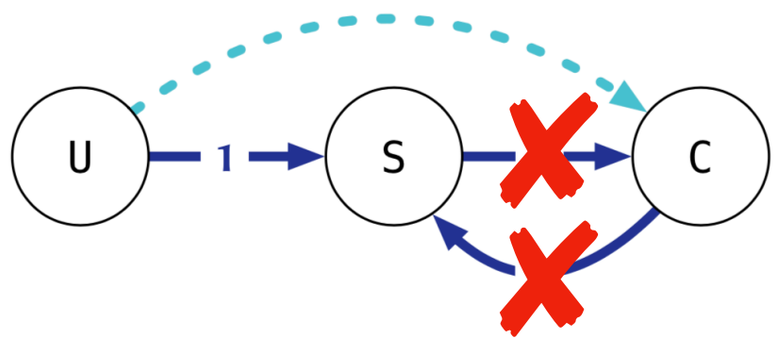
A	4
B	2
C	2
X	4

→

A	1
B	5
C	7
X	9

Fast estimation in external memory (TOIS 2019)





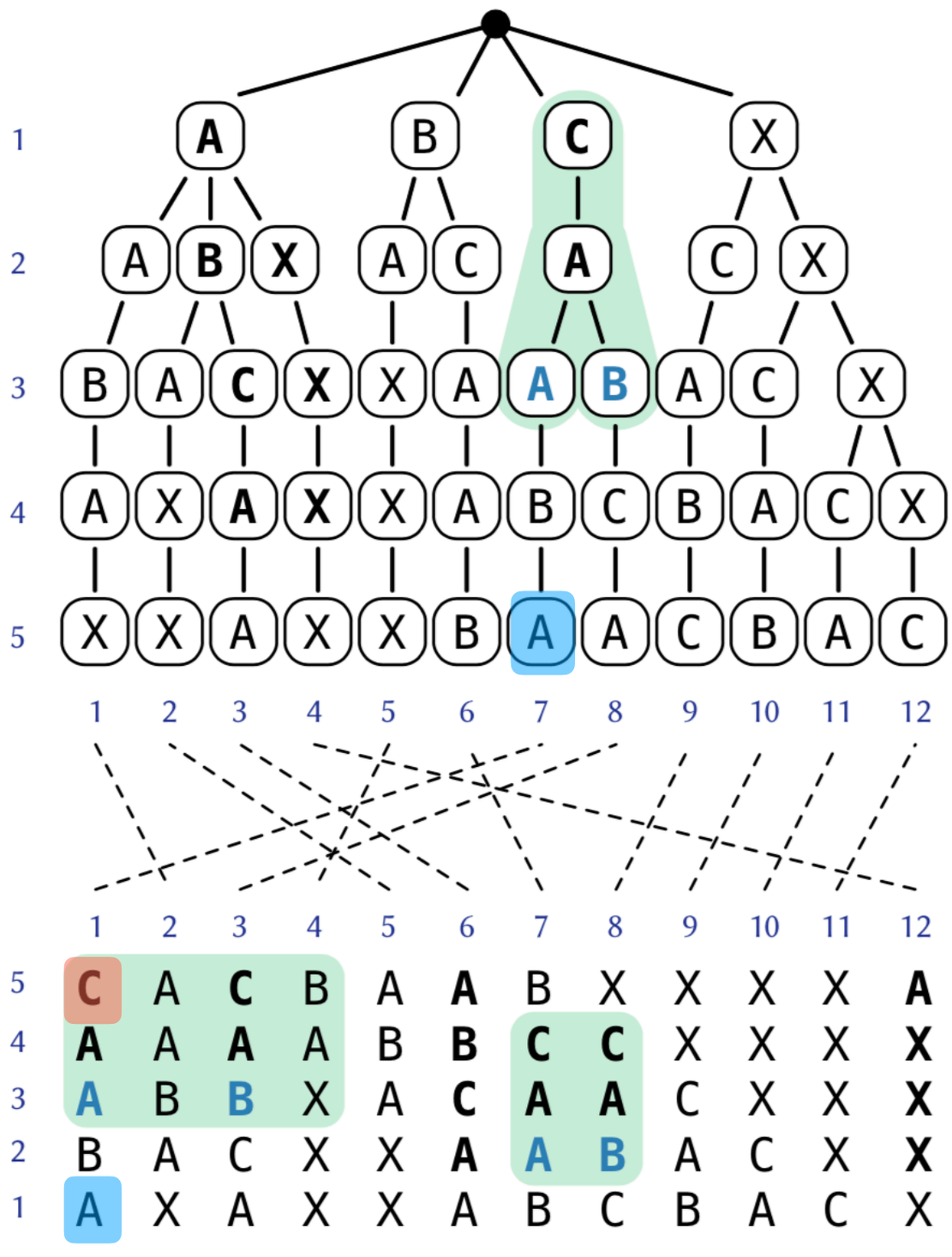
Rebuilding the last level of the trie.

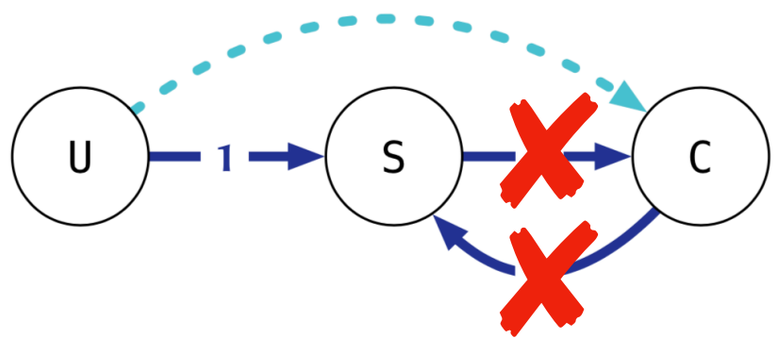
A	4
B	2
C	2
X	4

→

A	1
B	5
C	7
X	9

Fast estimation in external memory (TOIS 2019)





Rebuilding the last level of the trie.

A	4
B	2
C	2
X	4

→

A	1
B	5
C	7
X	9

Estimation runs 4.5X faster with billions of strings.

Take-home messages

- **Efficiency to deliver better services by using less resources. Impact is far reaching and implies substantial economic gains.**

- **Compression is mandatory if your data are “big”.**

- **Experiments are primary: design driven by numbers.**

Thanks for your attention,
time, patience!

Any questions?

High level thesis

Data Structures + Data Compression = Fast Algorithms

Design **space-efficient** *ad-hoc* data structures,
both from a theoretical *and* practical perspective,
that support **fast data extraction**.

Next word prediction

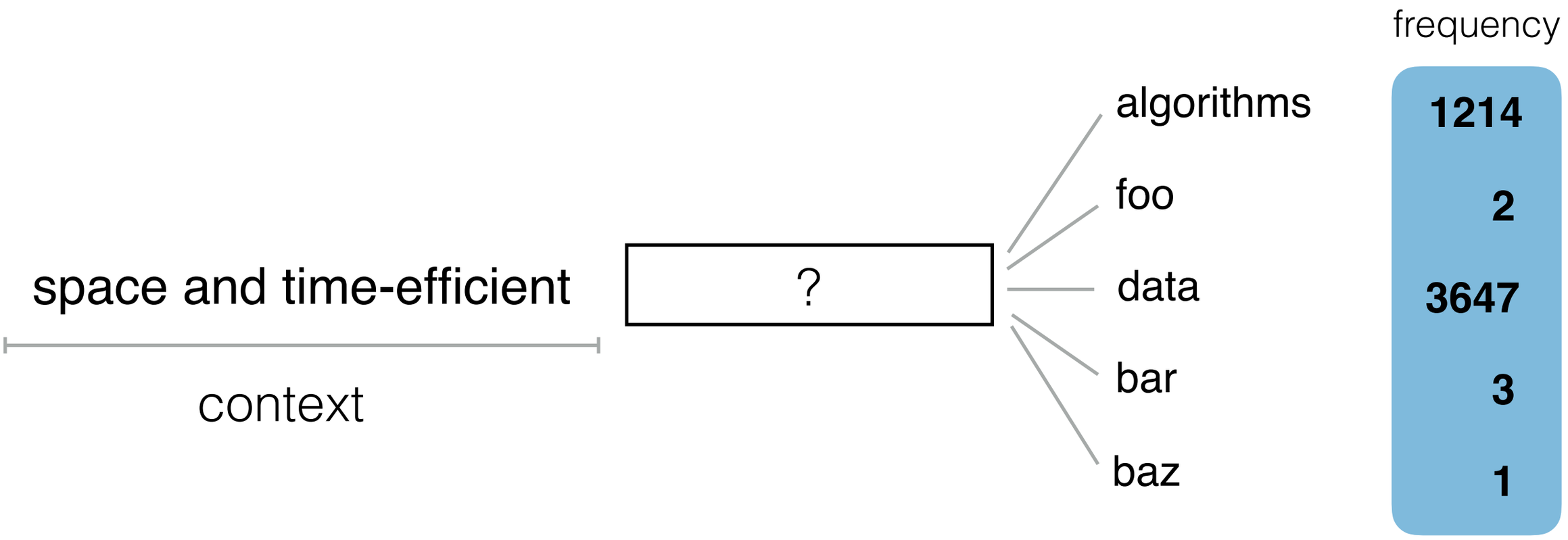
Next word prediction

space and time-efficient

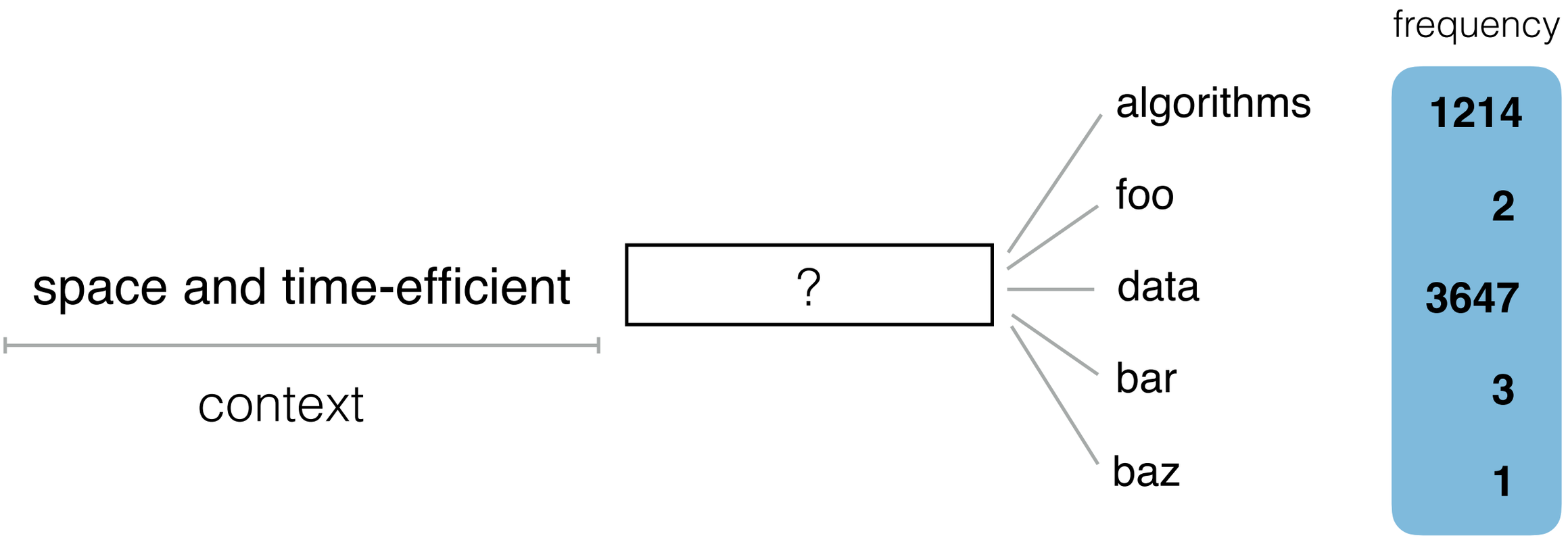
?

context

Next word prediction

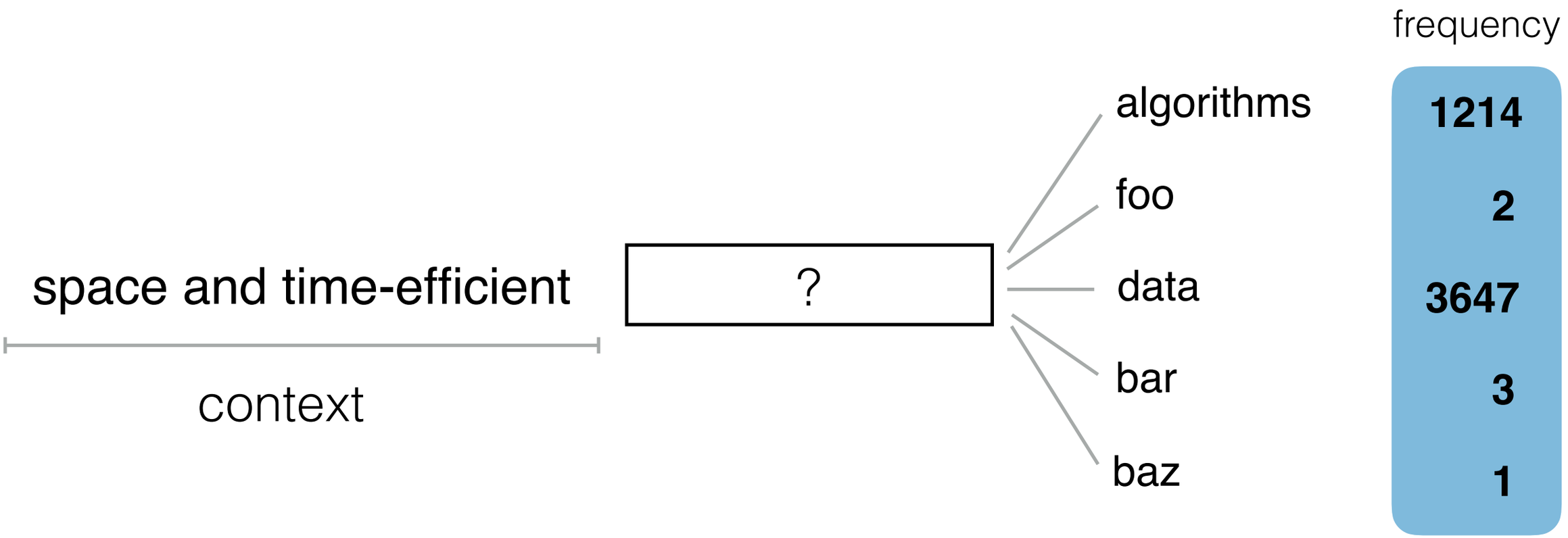


Next word prediction



$$\mathbf{P}(\text{“data” | “space and time-efficient”}) \approx \frac{\mathbf{f}(\text{“space and time-efficient data”})}{\mathbf{f}(\text{“space and time-efficient”})}$$

Next word prediction



$$\mathbf{P}(\text{“data”} \mid \text{“space and time-efficient”}) \approx \frac{\mathbf{f}(\text{“space and time-efficient data”})}{\mathbf{f}(\text{“space and time-efficient”})}$$

amazon alexa

Google Translate

Problem 3

Integer data structures

- van Emde Boas Trees
- X/Y-Fast Tries
- Fusion Trees
- Exponential Search Trees
- …

+ time
- space
+ dynamic

Elias-Fano encoding

- EF($S(n,u)$) = $n \log(u/n) + 2n$ bits to encode a sorted integer sequence S
- O(1) **Access**
- O(1 + $\log(u/n)$) **Predecessor**

+ time
+ space
- static

Problem 3

Integer data structures

- van Emde Boas Trees
- X/Y-Fast Tries
- Fusion Trees
- Exponential Search Trees
- …

Elias-Fano encoding

- EF($S(n,u)$) = $n \log(u/n) + 2n$ bits to encode a sorted integer sequence S
- O(1) **Access**
- O(1 + log(u/n)) **Predecessor**

+ time
- space
+ dynamic

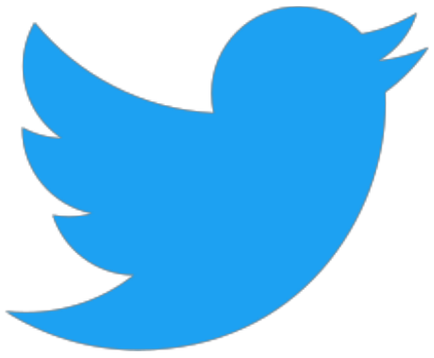
+ time
+ space
- static

Can we grab the best from both?

Dynamic inverted indexes

Classic solution: use two indexes.
One is big and **static**; the other is small and **dynamic**.
Merge them periodically.

Append-only inverted indexes.



Integer dictionaries in succinct space (CPM 2017)

Result 1

For $u = n^{\gamma}$, $\gamma = \Theta(1)$:

- **EF**($S(n,u)$) + o(n) bits
- O(1) **Access**
- O(min{1+log(u/n), loglog n}) **Predecessor**

Result 2

- **EF**($S(n,u)$) + o(n) bits
- O(1) **Access**
- O(1) **Append** (amortized)
- O(min{1+log(u/n), loglog n}) **Predecessor**

Result 3

- **EF**($S(n,u)$) + o(n) bits
- O(log n / loglog n) **Access**
- O(log n / loglog n) **Insert/Delete** (amortized)
- O(min{1+log(u/n), loglog n}) **Predecessor**

Integer dictionaries in succinct space (CPM 2017)

Result 1

For $u = n^\gamma$, $\gamma = \Theta(1)$:

- **EF**($S(n,u)$) + o(n) bits
- O(1) **Access**
- O(min{1+log(u/n), loglog n}) **Predecessor**

Result 2

- **EF**($S(n,u)$) + o(n) bits
- O(1) **Access**
- O(1) **Append** (amortized)
- O(min{1+log(u/n), loglog n}) **Predecessor**

Result 3

- **EF**($S(n,u)$) + o(n) bits
- O(log n / loglog n) **Access**
- O(log n / loglog n) **Insert/Delete** (amortized)
- O(min{1+log(u/n), loglog n}) **Predecessor**

Optimal time bounds for all operations using a sublinear redundancy.