Space and Time-Efficient Data Structures for Massive Datasets

Giulio Ermanno Pibiri
giulio.pibiri@di.unipi.it

Supervisor
Rossano Venturini

Computer Science Department
University of Pisa

10/10/2017

High Level Thesis

Data Structures + Data Compression ➡ Faster Algorithms

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

Data compression & Fast Retrieval
together.

Published Results

1. ***Clustered Elias-Fano Indexes***

2. ***Dynamic Elias-Fano Representation***

3. ***Efficient Data Structures for Massive N-Gram Datasets***

Published Results

1. ***Clustered Elias-Fano Indexes***

   Journal paper

   Giulio Ermanno Pibiri and Rossano Venturini
   ACM Transactions on Information Systems (TOIS), 2017

2. ***Dynamic Elias-Fano Representation***

3. ***Efficient Data Structures for Massive N-Gram Datasets***

Published Results

1. ***Clustered Elias-Fano Indexes*** **Journal paper**

   Giulio Ermanno Pibiri and Rossano Venturini
   ACM Transactions on Information Systems (TOIS), 2017

2. ***Dynamic Elias-Fano Representation*** **Conference paper**

   Giulio Ermanno Pibiri and Rossano Venturini
   Annual Symposium on Combinatorial Pattern Matching (CPM), 2017

3. ***Efficient Data Structures for Massive N-Gram Datasets***

Published Results

1. ***Clustered Elias-Fano Indexes*** **Journal paper**

   Giulio Ermanno Pibiri and Rossano Venturini
   ACM Transactions on Information Systems (TOIS), 2017

2. ***Dynamic Elias-Fano Representation*** **Conference paper**

   Giulio Ermanno Pibiri and Rossano Venturini
   Annual Symposium on Combinatorial Pattern Matching (CPM), 2017

3. ***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), 2017

Published Results

1. ***Clustered Elias-Fano Indexes*** **Journal paper**

   Giulio Ermanno Pibiri and Rossano Venturini
   ACM Transactions on Information Systems (TOIS), 2017

2. ***Dynamic Elias-Fano Representation*** **Conference paper**

   Giulio Ermanno Pibiri and Rossano Venturini
   Annual Symposium on Combinatorial Pattern Matching (CPM), 2017

3. ***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), 2017

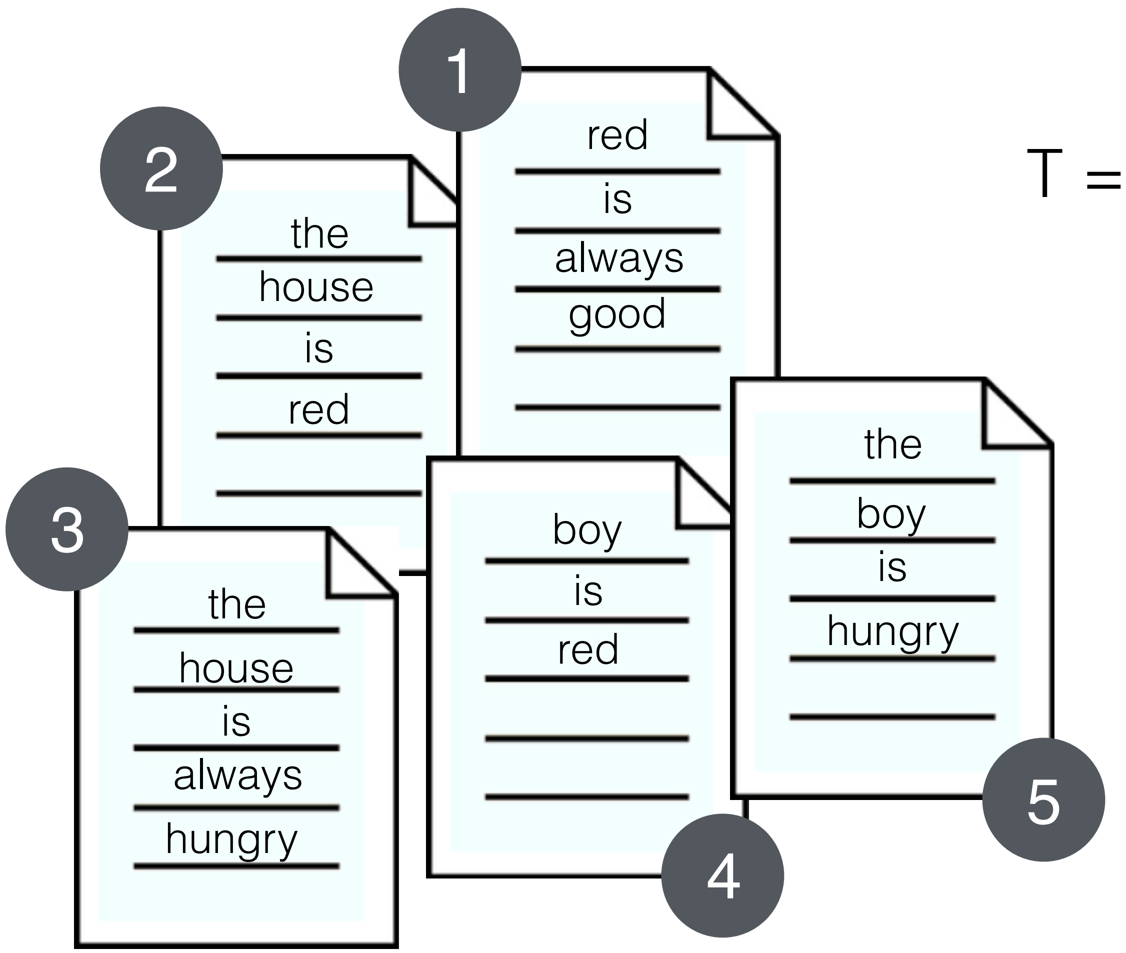
EVERYTHING that I do (papers, slides *and* **code**) is fully accessible at my page:
http://pages.di.unipi.it/pibiri/

Inverted Indexes

Inverted Indexes owe their popularity to the *efficient resolution of queries*, such as: “return me all documents in which terms $\{t_1,\ldots,t_k\}$ occur”.

Inverted Indexes

Inverted Indexes owe their popularity to the *efficient resolution of queries*, such as: “return me all documents in which terms $\{t_1,\ldots,t_k\}$ occur”.

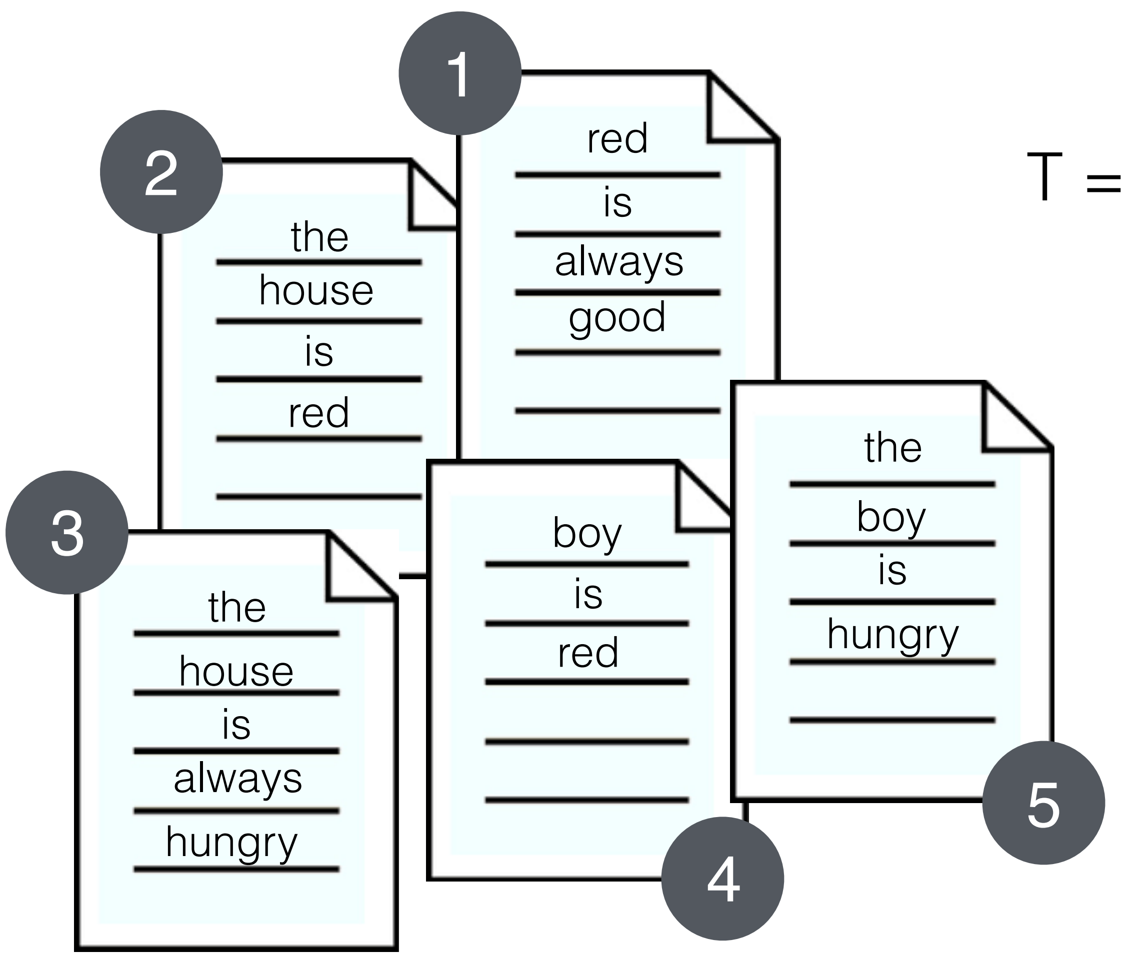


$T = \{\overset{t_1}{\text{always}}, \overset{t_2}{\text{boy}}, \overset{t_3}{\text{good}}, \overset{t_4}{\text{house}}, \overset{t_5}{\text{hungry}}, \overset{t_6}{\text{is}}, \overset{t_7}{\text{red}}, \overset{t_8}{\text{the}}\}$

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

Inverted Indexes

Inverted Indexes owe their popularity to the *efficient resolution of queries*, such as: “return me all documents in which terms $\{t_1,\ldots,t_k\}$ occur”.



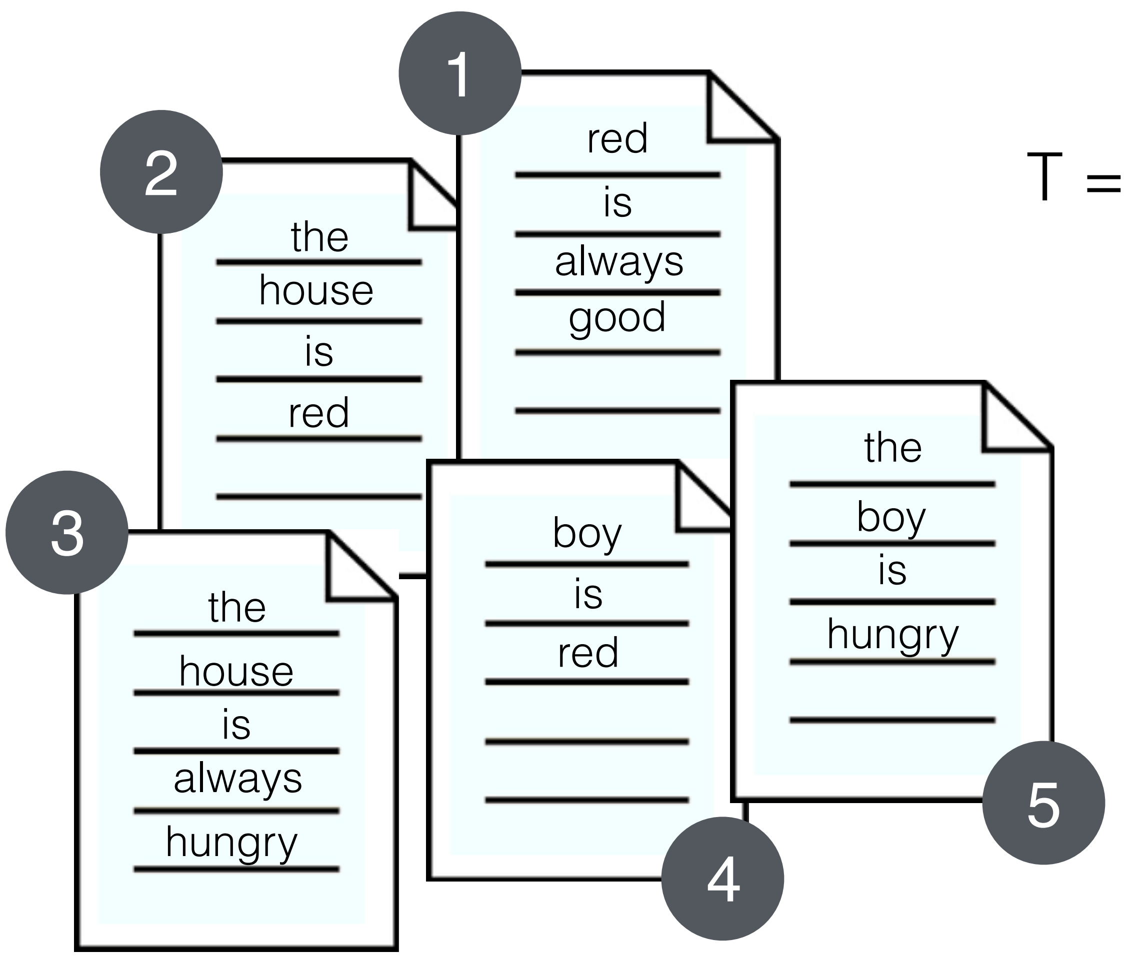
$$T = \{\overset{t_1}{\text{always}}, \overset{t_2}{\text{boy}}, \overset{t_3}{\text{good}}, \overset{t_4}{\text{house}}, \overset{t_5}{\text{hungry}}, \overset{t_6}{\text{is}}, \overset{t_7}{\text{red}}, \overset{t_8}{\text{the}}\}$$

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

$q = \{\text{boy}, \text{is}, \text{the}\}$

Inverted Indexes

Inverted Indexes owe their popularity to the *efficient resolution of queries*, such as: “return me all documents in which terms $\{t_1,\ldots,t_k\}$ occur”.



t_1 t_2 t_3 t_4 t_5 t_6 t_7 t_8

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

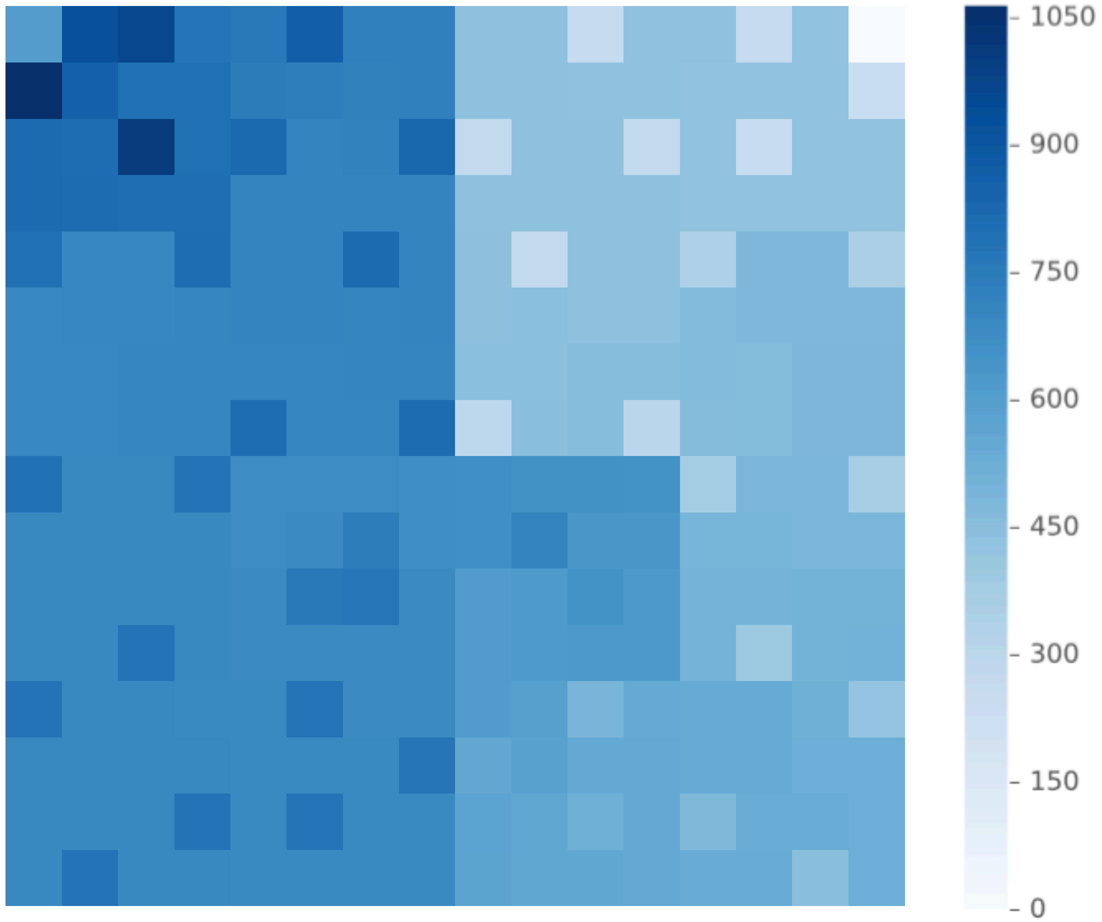
$L_{t_1}=[1, 3]$
$\boxed{L_{t_2}=[4, 5]}$
$L_{t_3}=[1]$
$L_{t_4}=[2, 3]$
$L_{t_5}=[3, 5]$
$\boxed{L_{t_6}=[1, 2, 3, 4, 5]}$
$L_{t_7}=[1, 2, 4]$
$\boxed{L_{t_8}=[2, 3, 5]}$

q = {boy, is, the}

Clustered Elias-Fano Indexes - TOIS'17

Every encoder represents each sequence *individually.*

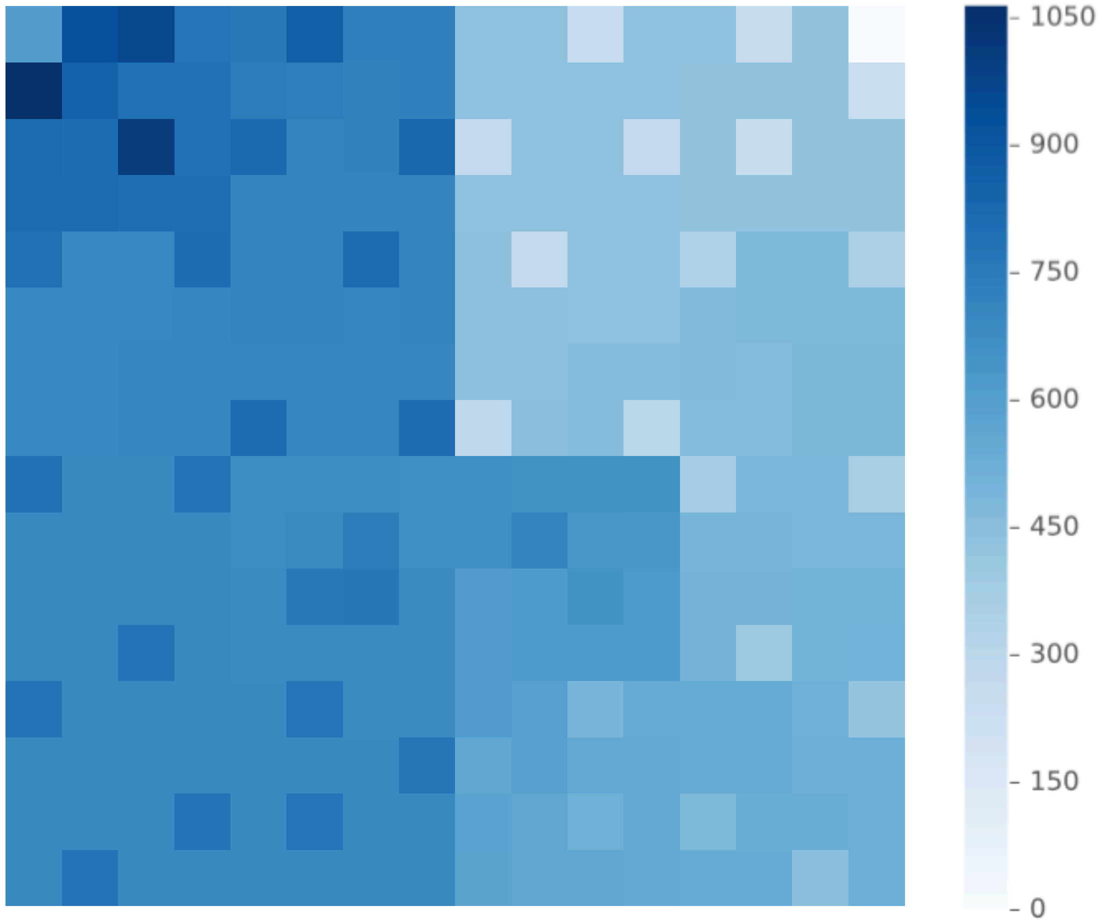
No exploitation of redundancy.



Clustered Elias-Fano Indexes - TOIS'17

Every encoder represents each sequence *individually.*

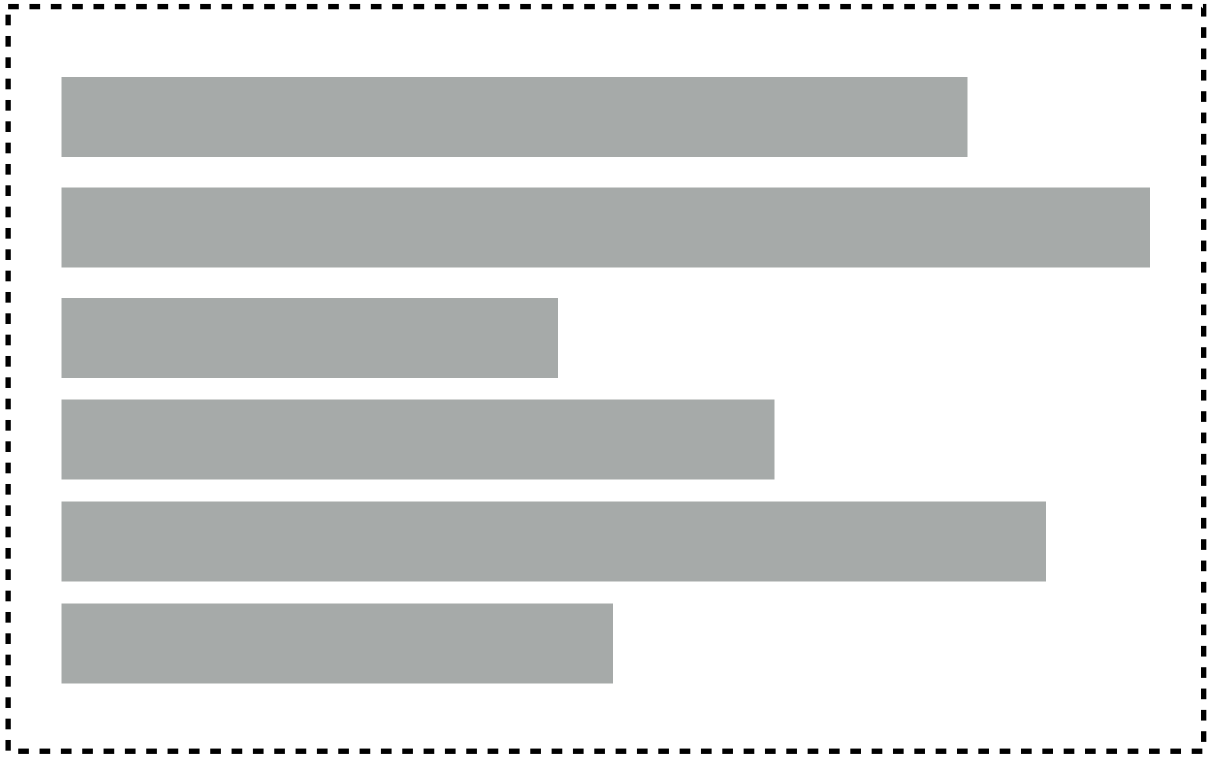
No exploitation of redundancy.



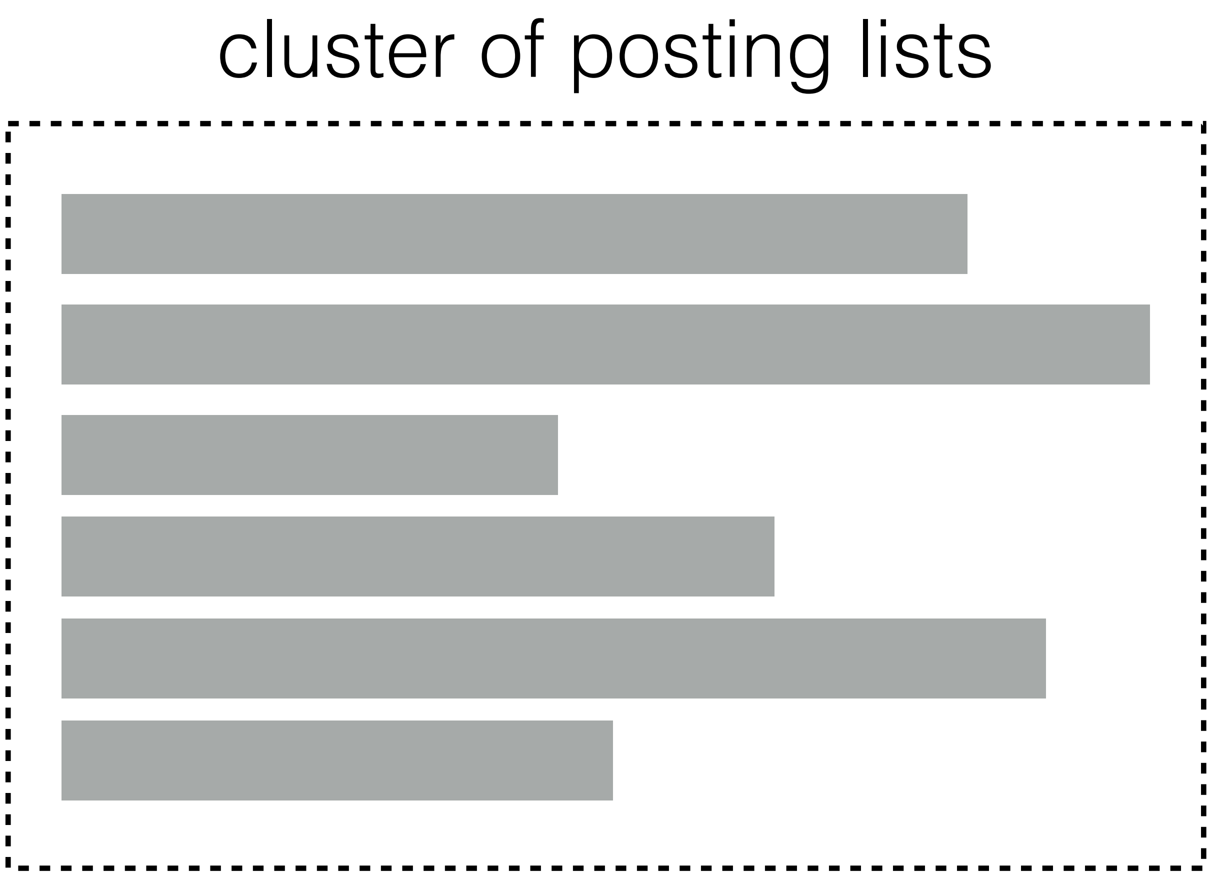
Idea: encode **clusters** of posting lists.

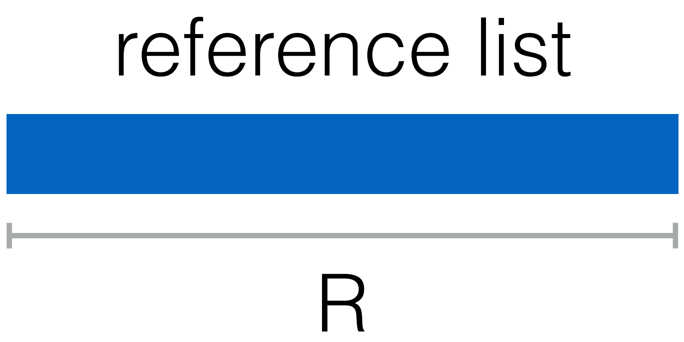
Clustered Elias-Fano Indexes - TOIS'17

cluster of posting lists

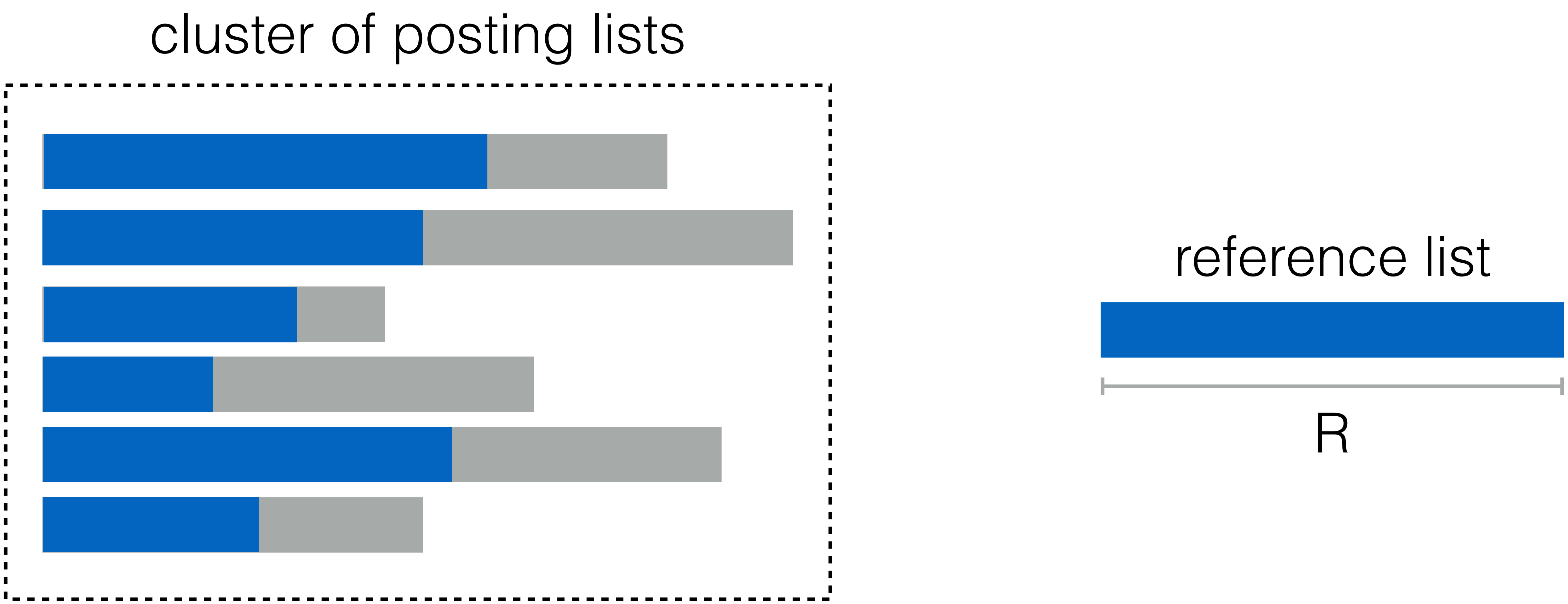


Clustered Elias-Fano Indexes - TOIS'17

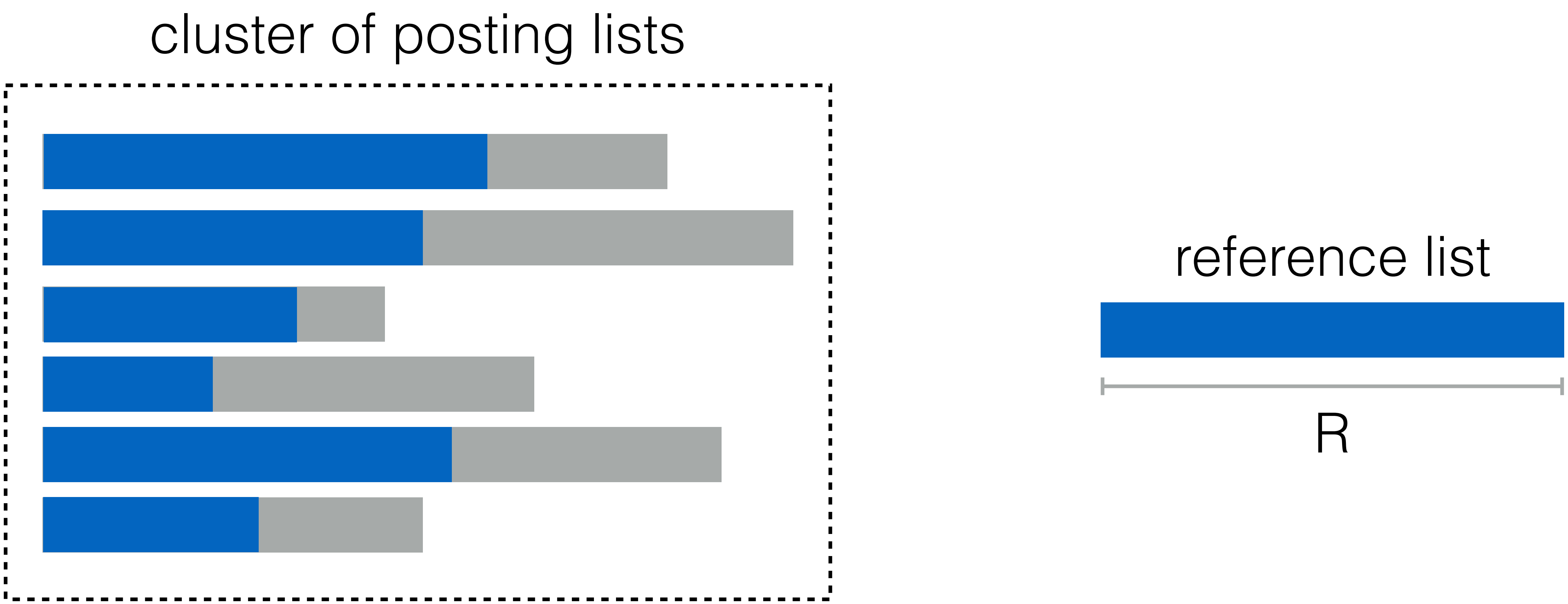




Clustered Elias-Fano Indexes - TOIS'17



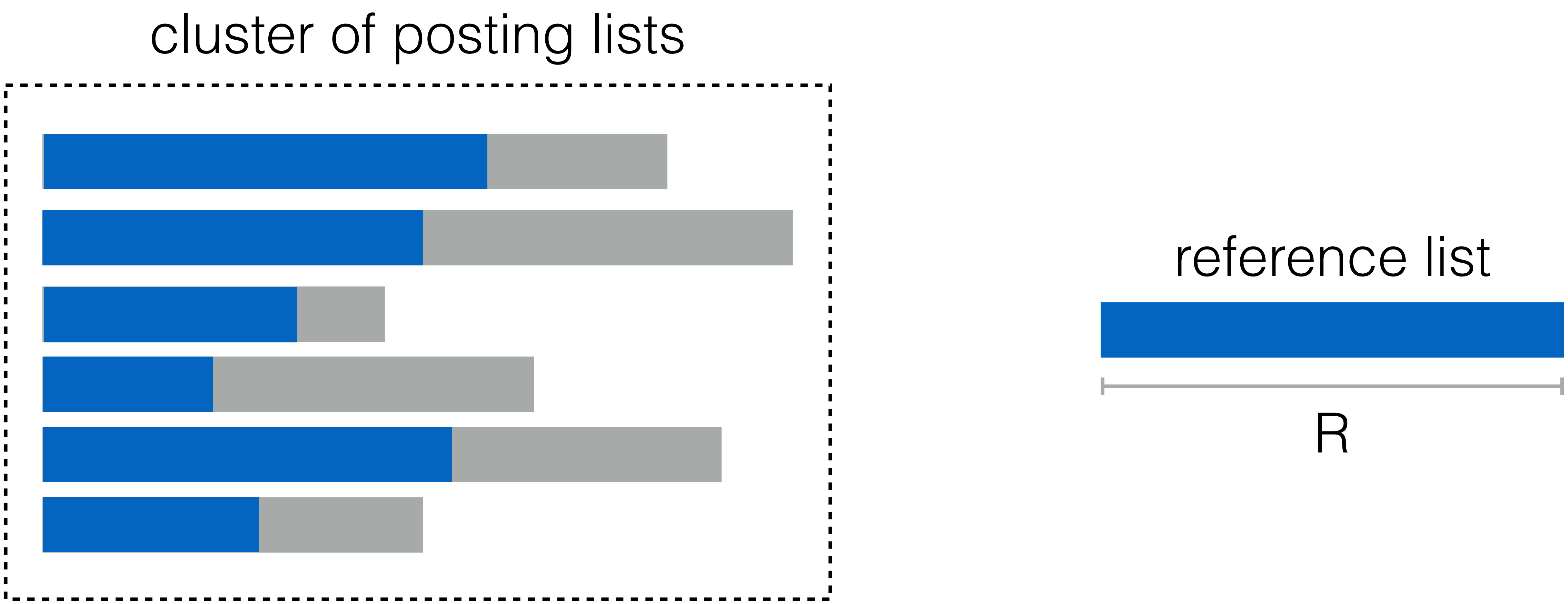
Clustered Elias-Fano Indexes - TOIS'17



R << u

log u bits VS **log R** bits

Clustered Elias-Fano Indexes - TOIS'17



R << u

log u bits

VS

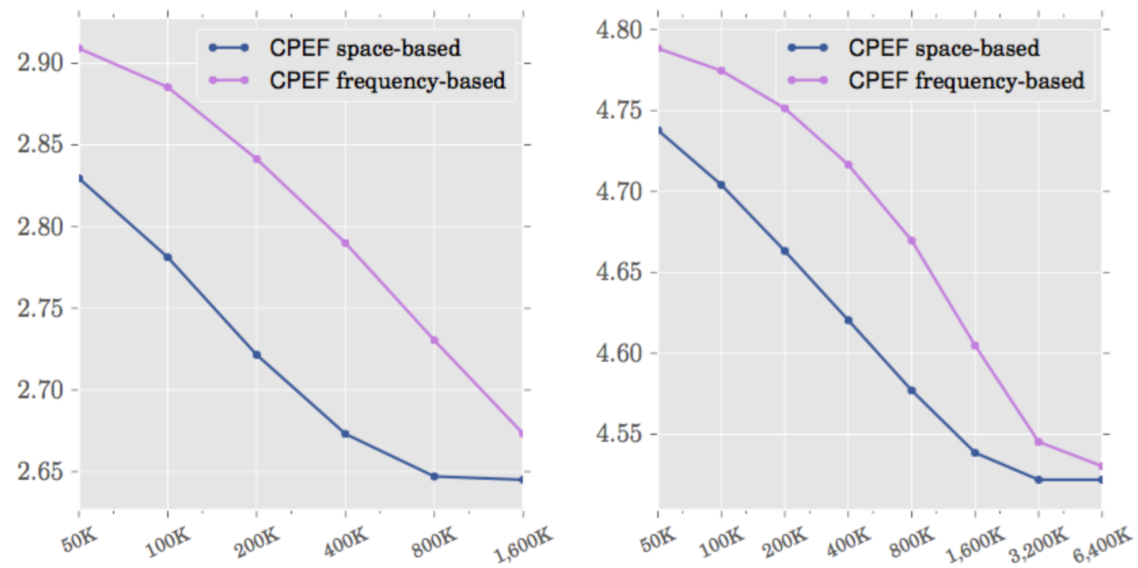
log R bits

Problems

1. Build the clusters.
2. Synthesise the reference list.

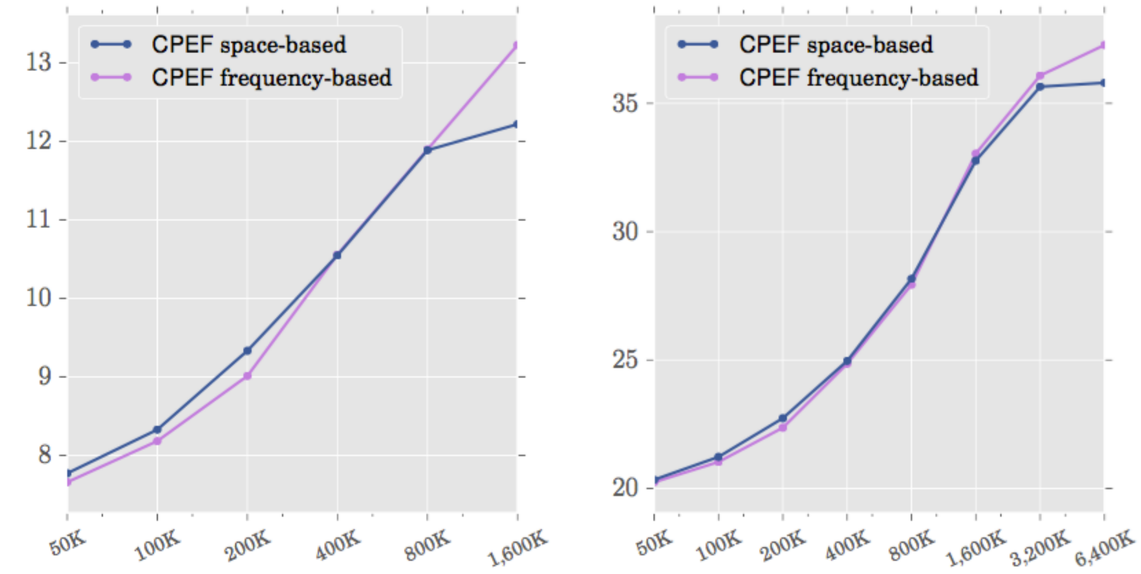
NP-hard problem
already for a simplified formulation.

Clustered Elias-Fano Indexes - TOIS'17



(a) Gov2 (b) ClueWeb09

Figure 2: Bits per posting of Gov2 and ClueWeb09 by varying the reference size.



(a) Gov2 (b) ClueWeb09

Figure 3: Timings for AND queries by varying the reference size on Gov2 and ClueWeb09, using the query set TREC 06.

	MIN		MID		MAX	
PEF	2.94	(+5.60%)	2.94	(+7.91%)	2.94	(+10.95%)
CPEF	2.78		2.72		2.65	
BIC	2.80	(+0.53%)	2.80	(+2.74%)	2.80	(+5.63%)

(a) Gov2

	MIN		MID		MAX	
PEF	4.80	(+2.13%)	4.80	(+3.98%)	4.80	(+6.25%)
CPEF	4.70		4.62		4.52	
BIC	4.27	(−9.22%)	4.27	(−7.58%)	4.27	(−5.56%)

(b) ClueWeb09

Table 2: Bits per posting in selected trade-off points.

		MIN		MID		MAX	
TREC 05	PEF	14.6	(−17.5%)	14.6	(−29.0%)	14.6	(−49.7%)
	CPEF	17.7		20.6		29.1	
	BIC	41.1	(+131.9%)	41.1	(+99.5%)	41.1	(+41.3%)
TREC 06	PEF	17.7	(−16.6%)	17.7	(−29.1%)	17.7	(−50.3%)
	CPEF	21.2		25.0		35.6	
	BIC	55.1	(+159.7%)	55.1	(+120.8%)	55.1	(+54.7%)

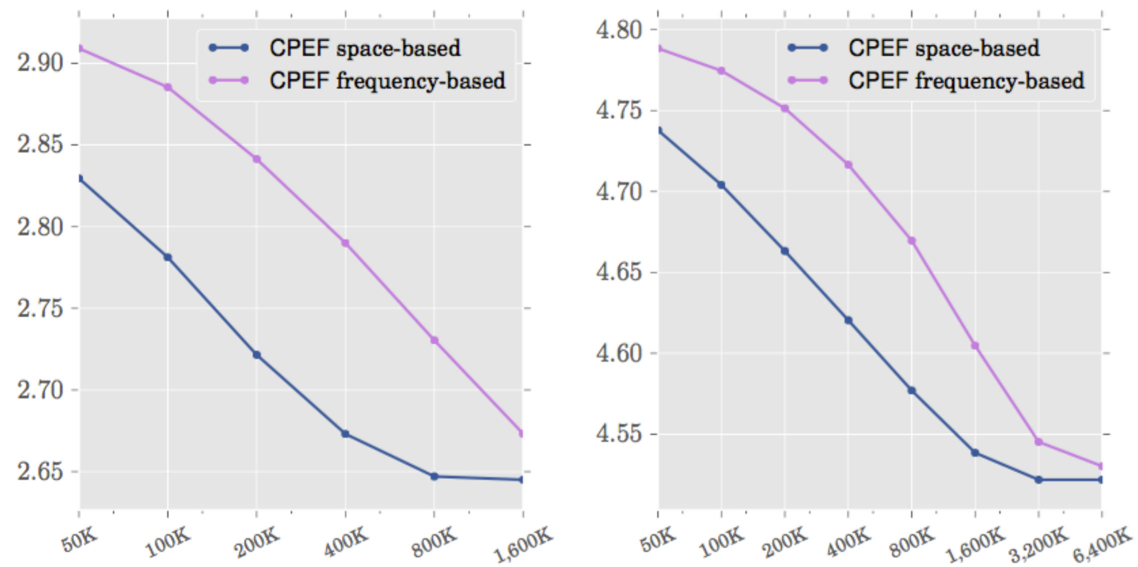
(a) ClueWeb09

		MIN		MID		MAX	
TREC 05	PEF	3.7	(−30.4%)	3.7	(−37.5%)	3.7	(−52.1%)
	CPEF	5.3		5.9		7.8	
	BIC	10.5	(+96.2%)	10.5	(+76.2%)	10.5	(+35.0%)
TREC 06	PEF	6.1	(−27.4%)	6.1	(−35.2%)	6.1	(−49.1%)
	CPEF	8.3		9.3		11.9	
	BIC	18.5	(+122.6%)	18.5	(+98.6%)	18.5	(+56.0%)

(b) Gov2

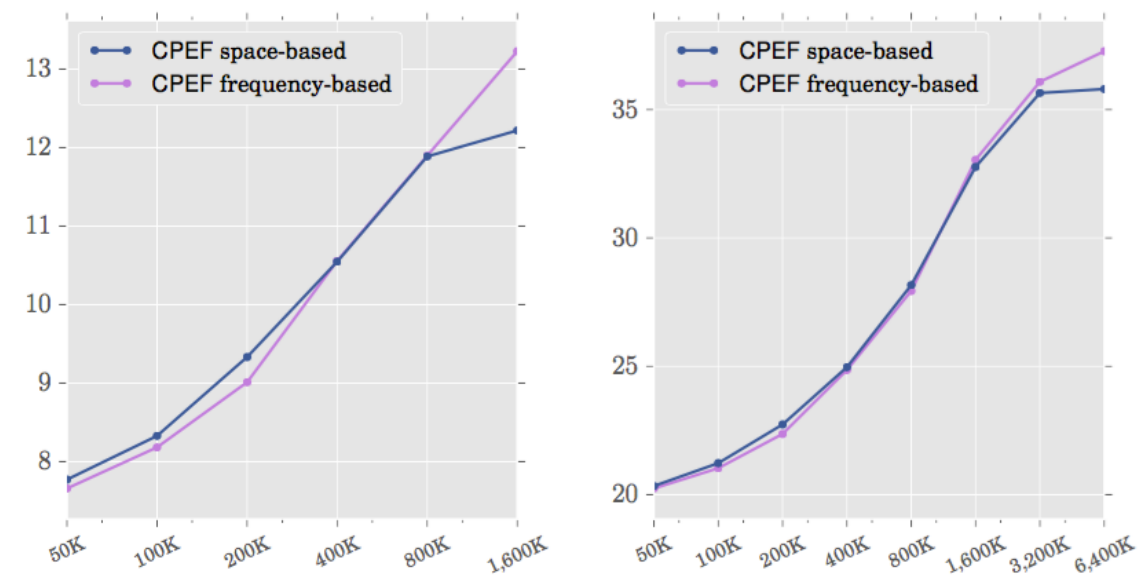
Table 3: Timings in milliseconds for AND queries on ClueWeb09 and Gov2, using query sets TREC 05 and TREC 05. In parentheses we show the relative percentage against CPEF.

Clustered Elias-Fano Indexes - TOIS'17



(a) Gov2 (b) ClueWeb09

Figure 2: Bits per posting of Gov2 and ClueWeb09 by varying the reference size.



(a) Gov2 (b) ClueWeb09

Figure 3: Timings for AND queries by varying the reference size on Gov2 and ClueWeb09, using the query set TREC 06.

	MIN		MID		MAX	
PEF	2.94	(+5.60%)	2.94	(+7.91%)	2.94	(+10.95%)
CPEF	2.78		2.72		2.65	
BIC	2.80	(+0.53%)	2.80	(+2.74%)	2.80	(+5.63%)

(a) Gov2

	MIN		MID		MAX	
PEF	4.80	(+2.13%)	4.80	(+3.98%)	4.80	(+6.25%)
CPEF	4.70		4.62		4.52	
BIC	4.27	(−9.22%)	4.27	(−7.58%)	4.27	(−5.56%)

(b) ClueWeb09

Table 2: Bits per posting in selected trade-off points.

		MIN		MID		MAX	
TREC 05	PEF	14.6	(−17.5%)	14.6	(−29.0%)	14.6	(−49.7%)
	CPEF	17.7		20.6		29.1	
	BIC	41.1	(+131.9%)	41.1	(+99.5%)	41.1	(+41.3%)
TREC 06	PEF	17.7	(−16.6%)	17.7	(−29.1%)	17.7	(−50.3%)
	CPEF	21.2		25.0		35.6	
	BIC	55.1	(+159.7%)	55.1	(+120.8%)	55.1	(+54.7%)

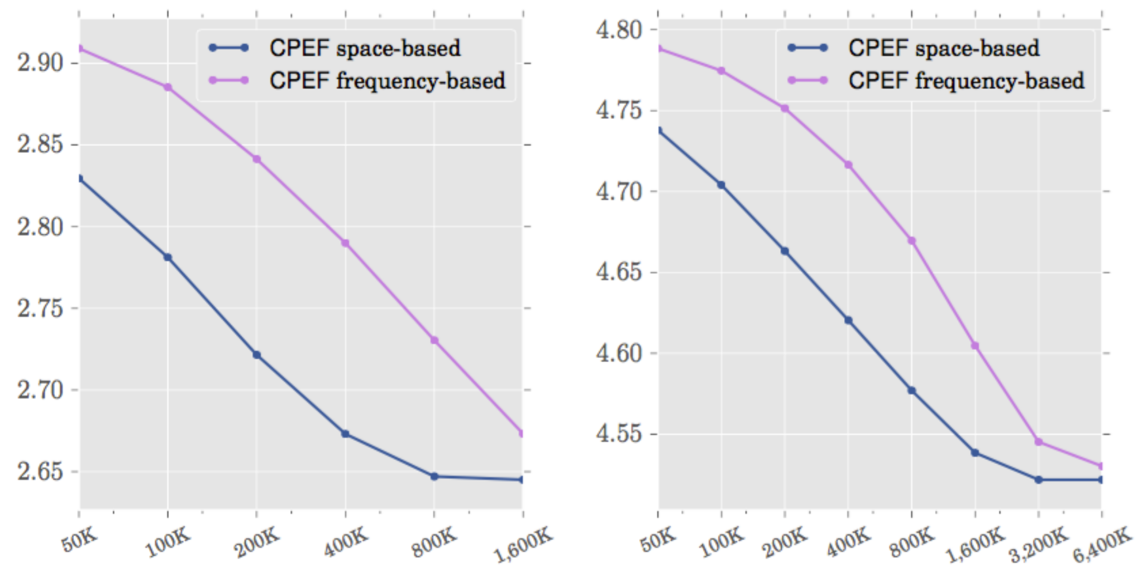
(a) ClueWeb09

		MIN		MID		MAX	
TREC 05	PEF	3.7	(−30.4%)	3.7	(−37.5%)	3.7	(−52.1%)
	CPEF	5.3		5.9		7.8	
	BIC	10.5	(+96.2%)	10.5	(+76.2%)	10.5	(+35.0%)
TREC 06	PEF	6.1	(−27.4%)	6.1	(−35.2%)	6.1	(−49.1%)
	CPEF	8.3		9.3		11.9	
	BIC	18.5	(+122.6%)	18.5	(+98.6%)	18.5	(+56.0%)

(b) Gov2

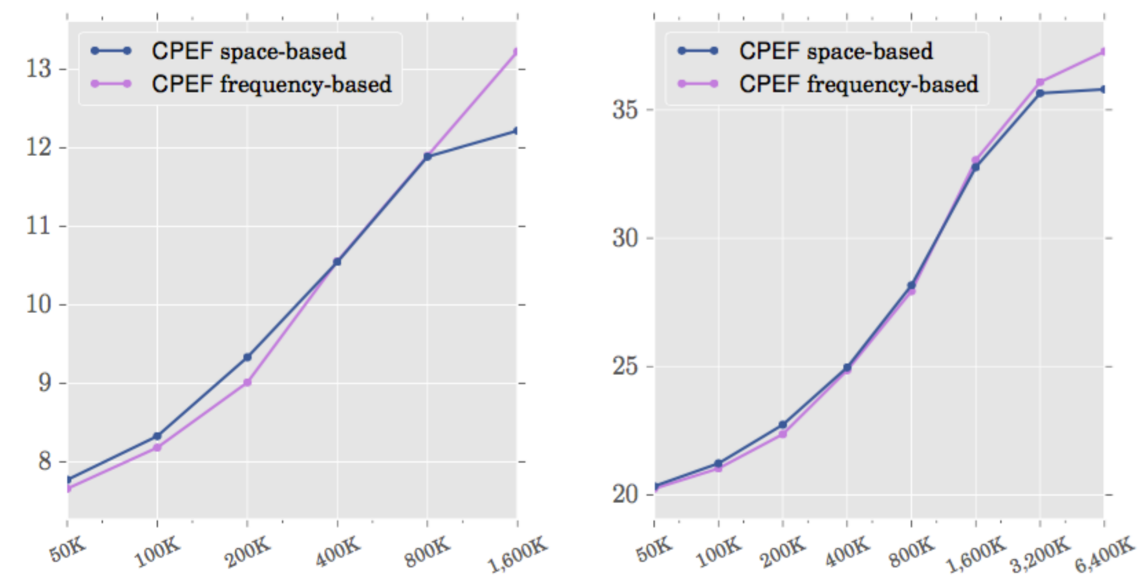
Table 3: Timings in milliseconds for AND queries on ClueWeb09 and Gov2, using query sets TREC 05 and TREC 05. In parentheses we show the relative percentage against CPEF.

Clustered Elias-Fano Indexes - TOIS'17



(a) Gov2 (b) ClueWeb09

Figure 2: Bits per posting of Gov2 and ClueWeb09 by varying the reference size.



(a) Gov2 (b) ClueWeb09

Figure 3: Timings for AND queries by varying the reference size on Gov2 and ClueWeb09, using the query set TREC 06.

	MIN		MID		MAX	
PEF	2.94	(+5.60%)	2.94	(+7.91%)	2.94	(+10.95%)
CPEF	2.78		2.72		2.65	
BIC	2.80	(+0.53%)	2.80	(+2.74%)	2.80	(+5.63%)

(a) Gov2

	MIN		MID		MAX	
PEF	4.80	(+2.13%)	4.80	(+3.98%)	4.80	(+6.25%)
CPEF	4.70		4.62		4.52	
BIC	4.27	(−9.22%)	4.27	(−7.58%)	4.27	(−5.56%)

(b) ClueWeb09

Table 2: Bits per posting in selected trade-off points.

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

		MIN		MID		MAX	
TREC 05	PEF	14.6	(−17.5%)	14.6	(−29.0%)	14.6	(−49.7%)
	CPEF	17.7		20.6		29.1	
	BIC	41.1	(+131.9%)	41.1	(+99.5%)	41.1	(+41.3%)
TREC 06	PEF	17.7	(−16.6%)	17.7	(−29.1%)	17.7	(−50.3%)
	CPEF	21.2		25.0		35.6	
	BIC	55.1	(+159.7%)	55.1	(+120.8%)	55.1	(+54.7%)

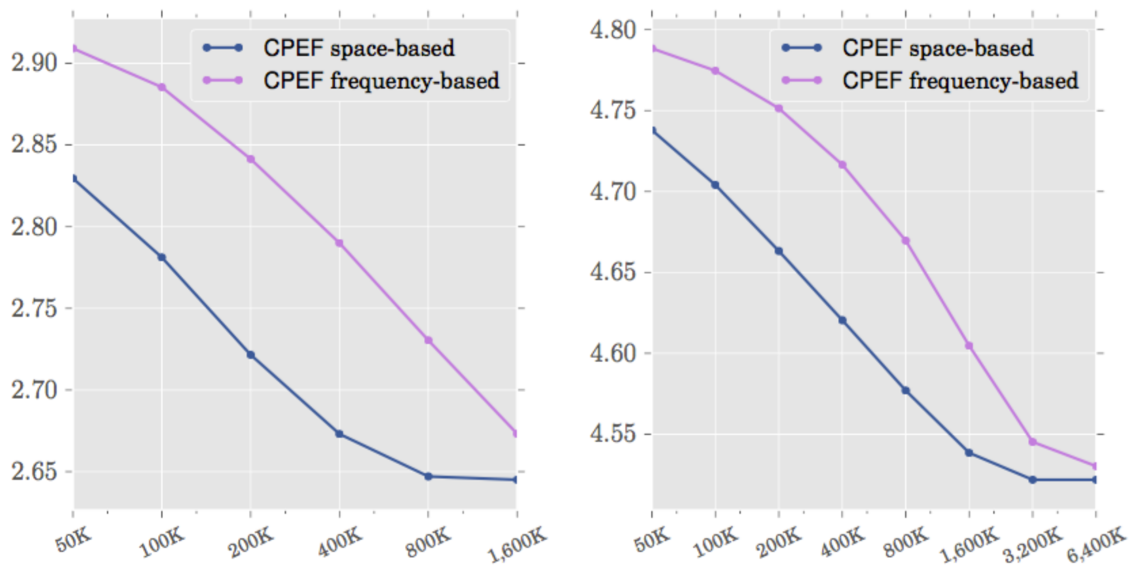
(a) ClueWeb09

		MIN		MID		MAX	
TREC 05	PEF	3.7	(−30.4%)	3.7	(−37.5%)	3.7	(−52.1%)
	CPEF	5.3		5.9		7.8	
	BIC	10.5	(+96.2%)	10.5	(+76.2%)	10.5	(+35.0%)
TREC 06	PEF	6.1	(−27.4%)	6.1	(−35.2%)	6.1	(−49.1%)
	CPEF	8.3		9.3		11.9	
	BIC	18.5	(+122.6%)	18.5	(+98.6%)	18.5	(+56.0%)

(b) Gov2

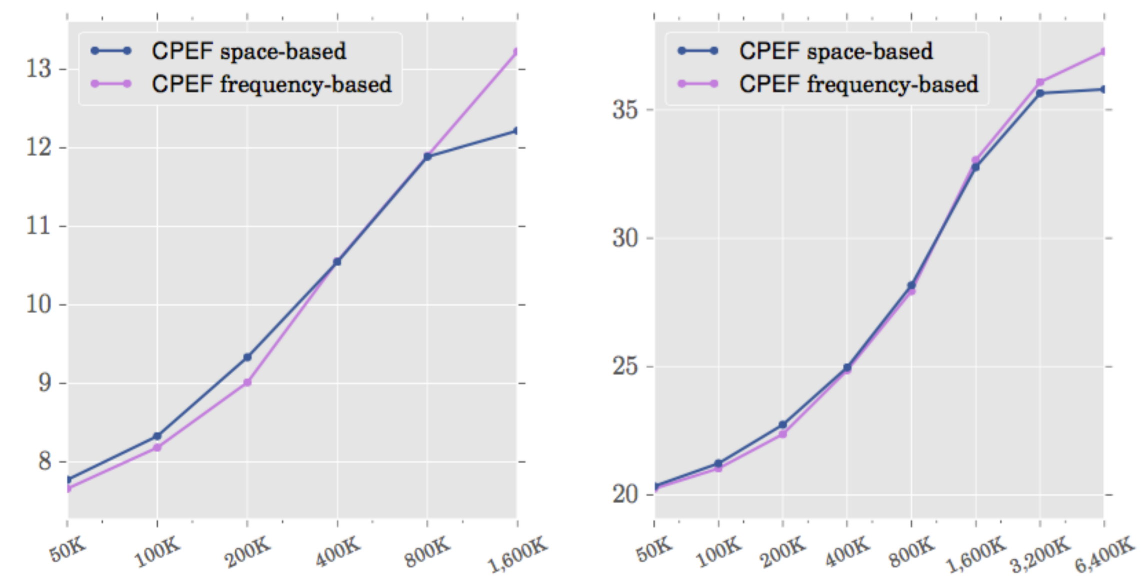
Table 3: Timings in milliseconds for AND queries on ClueWeb09 and Gov2, using query sets TREC 05 and TREC 05. In parentheses we show the relative percentage against CPEF.

Clustered Elias-Fano Indexes - TOIS'17



(a) Gov2 (b) ClueWeb09

Figure 2: Bits per posting of Gov2 and ClueWeb09 by varying the reference size.



(a) Gov2 (b) ClueWeb09

Figure 3: Timings for AND queries by varying the reference size on Gov2 and ClueWeb09, using the query set TREC 06.

	MIN		MID		MAX	
PEF	2.94	(+5.60%)	2.94	(+7.91%)	2.94	(+10.95%)
CPEF	2.78		2.72		2.65	
BIC	2.80	(+0.53%)	2.80	(+2.74%)	2.80	(+5.63%)

(a) Gov2

	MIN		MID		MAX	
PEF	4.80	(+2.13%)	4.80	(+3.98%)	4.80	(+6.25%)
CPEF	4.70		4.62		4.52	
BIC	4.27	(−9.22%)	4.27	(−7.58%)	4.27	(−5.56%)

(b) ClueWeb09

Table 2: Bits per posting in selected trade-off points.

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

		MIN		MID		MAX	
TREC 05	PEF	14.6	(−17.5%)	14.6	(−29.0%)	14.6	(−49.7%)
	CPEF	17.7		20.6		29.1	
	BIC	41.1	(+131.9%)	41.1	(+99.5%)	41.1	(+41.3%)
TREC 06	PEF	17.7	(−16.6%)	17.7	(−29.1%)	17.7	(−50.3%)
	CPEF	21.2		25.0		35.6	
	BIC	55.1	(+159.7%)	55.1	(+120.8%)	55.1	(+54.7%)

(a) ClueWeb09

		MIN		MID		MAX	
TREC 05	PEF	3.7	(−30.4%)	3.7	(−37.5%)	3.7	(−52.1%)
	CPEF	5.3		5.9		7.8	
	BIC	10.5	(+96.2%)	10.5	(+76.2%)	10.5	(+35.0%)
TREC 06	PEF	6.1	(−27.4%)	6.1	(−35.2%)	6.1	(−49.1%)
	CPEF	8.3		9.3		11.9	
	BIC	18.5	(+122.6%)	18.5	(+98.6%)	18.5	(+56.0%)

(b) Gov2

Table 3: Timings in milliseconds for AND queries on ClueWeb09 and Gov2, using query sets TREC 05 and TREC 05. In parentheses we show the relative percentage against CPEF.

Clustered Elias-Fano Indexes - TOIS’17

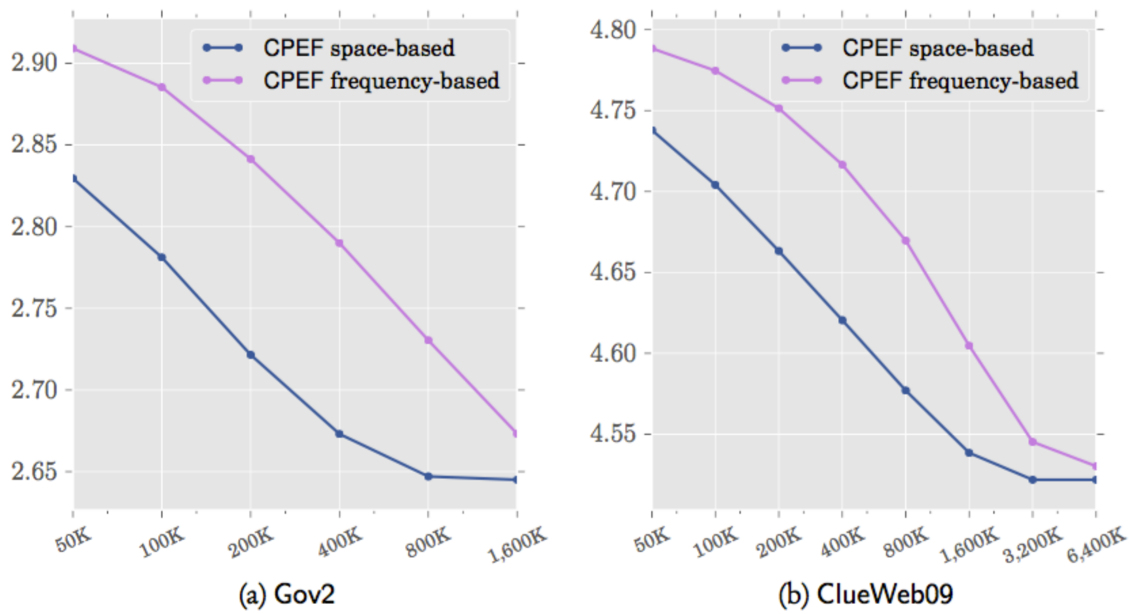


Figure 2: Bits per posting of Gov2 and ClueWeb09 by varying the reference size.

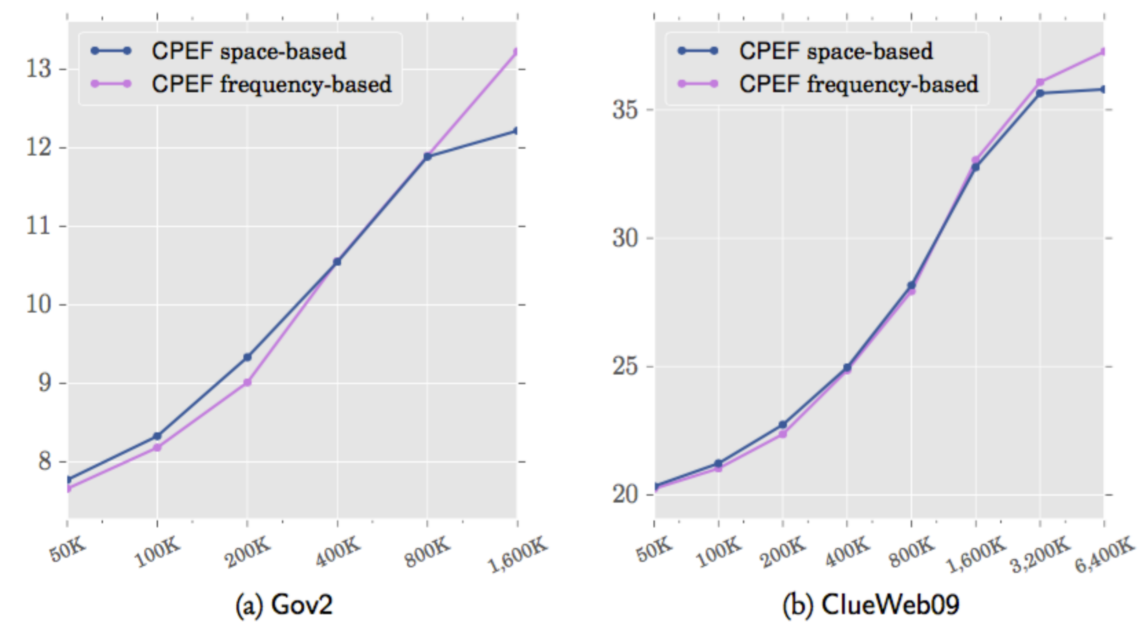


Figure 3: Timings for AND queries by varying the reference size on Gov2 and ClueWeb09, using the query set TREC 06.

	MIN	MID	MAX
PEF	2.94 (+5.60%)	2.94 (+7.91%)	2.94 (+10.95%)
CPEF	2.78	2.72	2.65
BIC	2.80 (+0.53%)	2.80 (+2.74%)	2.80 (+5.63%)

(a) Gov2

	MIN	MID	MAX
PEF	4.80 (+2.13%)	4.80 (+3.98%)	4.80 (+6.25%)
CPEF	4.70	4.62	4.52
BIC	4.27 (−9.22%)	4.27 (−7.58%)	4.27 (−5.56%)

(b) ClueWeb09

Table 2: Bits per posting in selected trade-off points.

		MIN	MID	MAX
TREC 05	PEF	14.6 (−17.5%)	14.6 (−29.0%)	14.6 (−49.7%)
TREC 05	CPEF	17.7	20.6	29.1
TREC 05	BIC	41.1 (+131.9%)	41.1 (+99.5%)	41.1 (+41.3%)
TREC 06	PEF	17.7 (−16.6%)	17.7 (−29.1%)	17.7 (−50.3%)
TREC 06	CPEF	21.2	25.0	35.6
TREC 06	BIC	55.1 (+159.7%)	55.1 (+120.8%)	55.1 (+54.7%)

(a) ClueWeb09

		MIN	MID	MAX
TREC 05	PEF	3.7 (−30.4%)	3.7 (−37.5%)	3.7 (−52.1%)
TREC 05	CPEF	5.3	5.9	7.8
TREC 05	BIC	10.5 (+96.2%)	10.5 (+76.2%)	10.5 (+35.0%)
TREC 06	PEF	6.1 (−27.4%)	6.1 (−35.2%)	6.1 (−49.1%)
TREC 06	CPEF	8.3	9.3	11.9
TREC 06	BIC	18.5 (+122.6%)	18.5 (+98.6%)	18.5 (+56.0%)

(b) Gov2

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

Much faster than BIC (103% on average)
Slightly slower than PEF (20% on average)

(Integer) Dynamic Ordered Sets

A **dynamic ordered set** S is a data structure representing n keys and supporting the following operations:

- $\mathsf{Insert}(x)$ inserts x in S
- $\mathsf{Delete}(x)$ deletes x from S
- $\mathsf{Search}(x)$ checks whether x belongs to S
- $\mathsf{Minimum}()$ returns the minimum element of S
- $\mathsf{Maximum}()$ returns the maximum element of S
- $\mathsf{Predecessor}(x)$ returns $\max\{y \in S : y < x\}$
- $\mathsf{Successor}(x)$ returns $\min\{y \in S : y \geq x\}$

(Integer) Dynamic Ordered Sets

A **dynamic ordered set** S is a data structure representing n keys and supporting the following operations:

- Insert(x) inserts x in S
- Delete(x) deletes x from S
- Search(x) checks whether x belongs to S
- Minimum() returns the minimum element of S
- Maximum() returns the maximum element of S
- Predecessor(x) returns $\max\{y \in S : y < x\}$
- Successor(x) returns $\min\{y \in S : y \geq x\}$

In the **comparison model** this is solved optimally by any self-balancing tree data structure in $O(\log n)$ time and $O(n)$ space.

More efficient solutions there exist if the considered keys are **integers** drawn from a bounded universe of size u.

(Integer) Dynamic Ordered Sets

A **dynamic ordered set** S is a data structure representing n keys and supporting the following operations:

- Insert(x) inserts x in S
- Delete(x) deletes x from S
- Search(x) checks whether x belongs to S
- Minimum() returns the minimum element of S
- Maximum() returns the maximum element of S
- Predecessor(x) returns $\max\{y \in S : y < x\}$
- Successor(x) returns $\min\{y \in S : y \geq x\}$

In the **comparison model** this is solved optimally by any self-balancing tree data structure in $O(\log n)$ time and $O(n)$ space.

More efficient solutions there exist if the considered keys are **integers** drawn from a bounded universe of size u.

Challenge

How to **optimally** solve the **integer** dynamic ordered set problem in **compressed space**?

Motivation

Integer Data Structures

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

\+ time
\- space
\+ dynamic

Elias-Fano Encoding

- $\text{EF}(S(n,u)) = n \log(u/n) + 2n$ bits to encode an ordered integer sequence S
- $O(1)$ **Access**
- $O(1 + \log(u/n))$ **Predecessor**

\+ time
\+ space
\- static

Motivation

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 an ordered integer sequence S
- O(1) **Access**
- O(1 + $\log(u/n)$) **Predecessor**

\+ time
\- space
\+ dynamic

\+ time
\+ space
\- static

Motivation

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 an ordered 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 Elias-Fano Representation - CPM'17

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 1

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

Result 2

- 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

Result 3

Dynamic Elias-Fano Representation - CPM'17

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 1

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

Result 2

- 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

Result 3

Dynamic Elias-Fano Representation - CPM'17

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 1

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

Result 2

- 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

Result 3

Dynamic Elias-Fano Representation - CPM'17

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 1

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

Result 2

- 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

Result 3

Dynamic Elias-Fano Representation - CPM'17

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 1

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

Result 2

- 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

Result 3

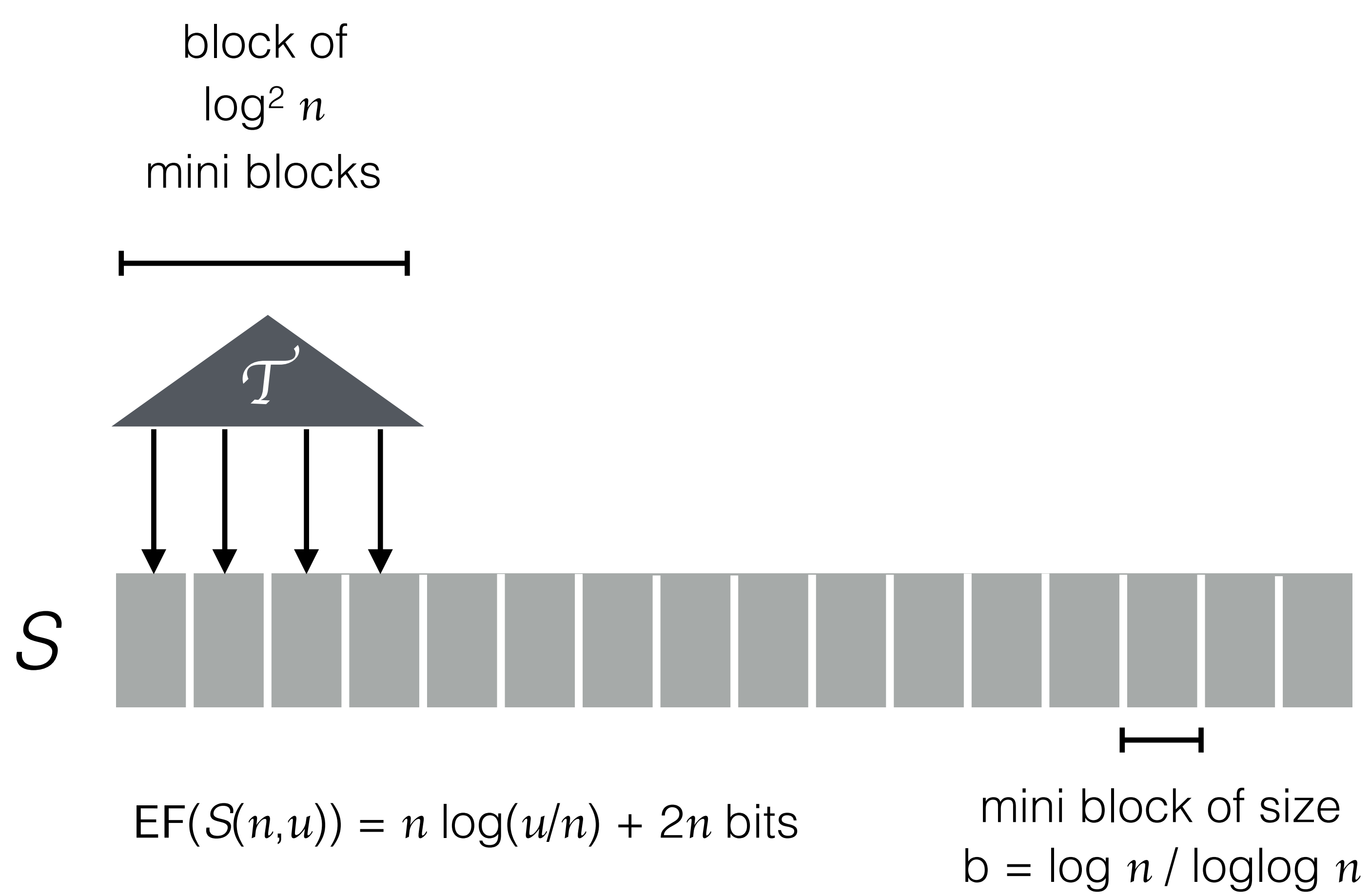
Dynamic Elias-Fano Representation - CPM'17

S

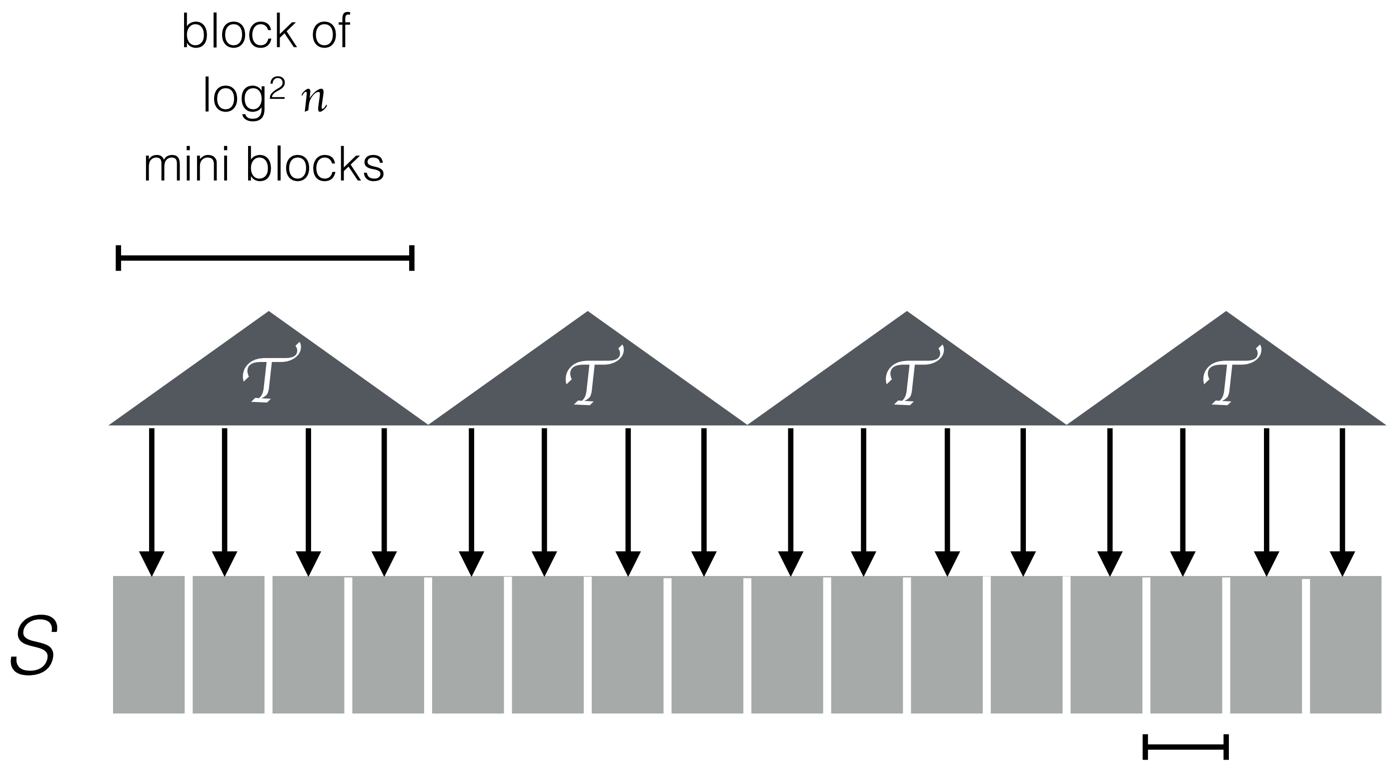
EF($S(n,u)$) = $n \log(u/n) + 2n$ bits

mini block of size
b = log n / loglog n

Dynamic Elias-Fano Representation - CPM'17



Dynamic Elias-Fano Representation - CPM'17

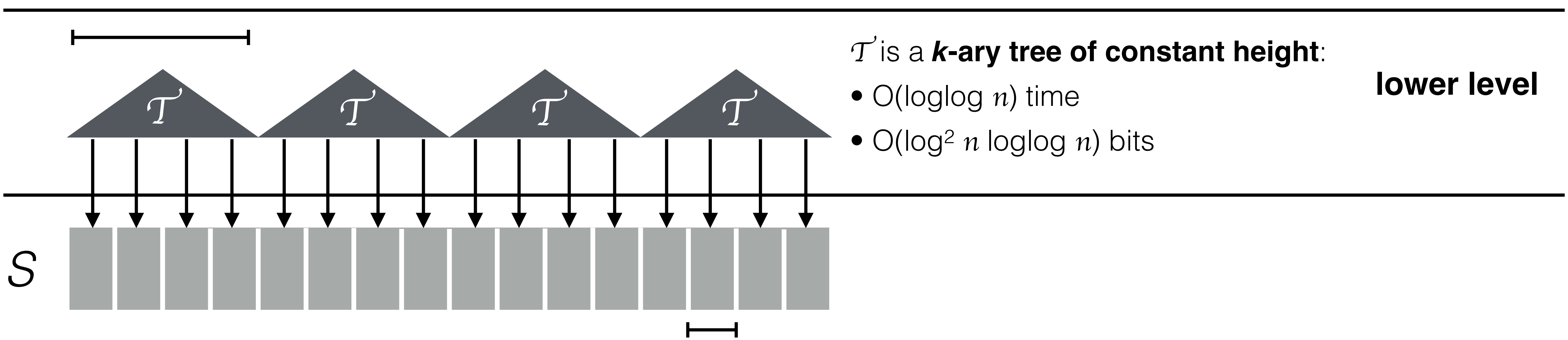


EF($S(n,u)$) = $n \log(u/n) + 2n$ bits

mini block of size
b = $\log n$ / $\log\log n$

Dynamic Elias-Fano Representation - CPM'17

block of
$\log^2 n$
mini blocks



$\mathcal{T}$ is a ***k*-ary tree of constant height**:

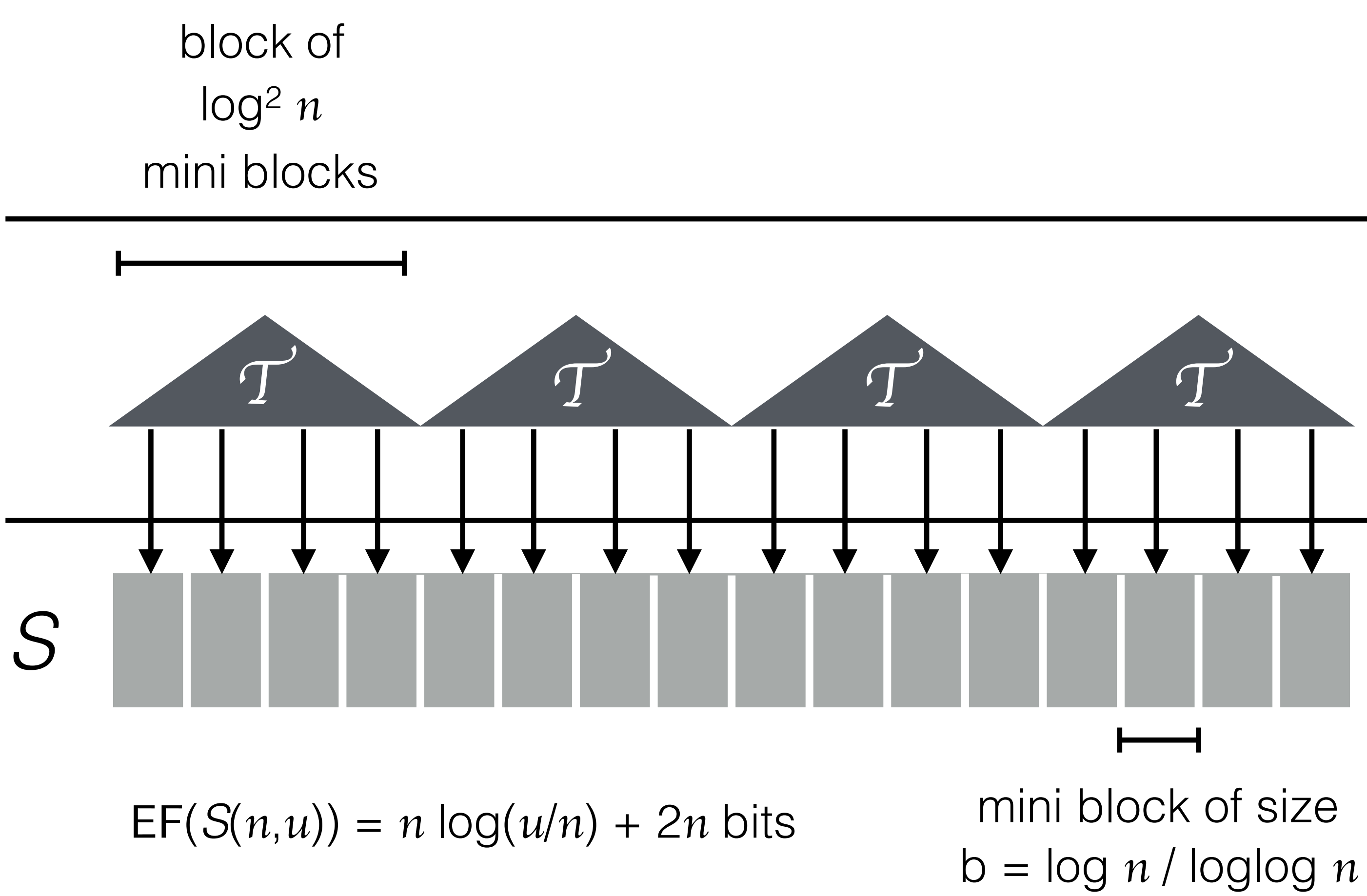
- $O(\log\log n)$ time
- $O(\log^2 n \log\log n)$ bits

lower level

EF($S(n,u)$) = $n \log(u/n) + 2n$ bits

mini block of size
b = $\log n$ / $\log\log n$

Dynamic Elias-Fano Representation - CPM'17



EF($S(n,u)$) = $n \log(u/n) + 2n$ bits

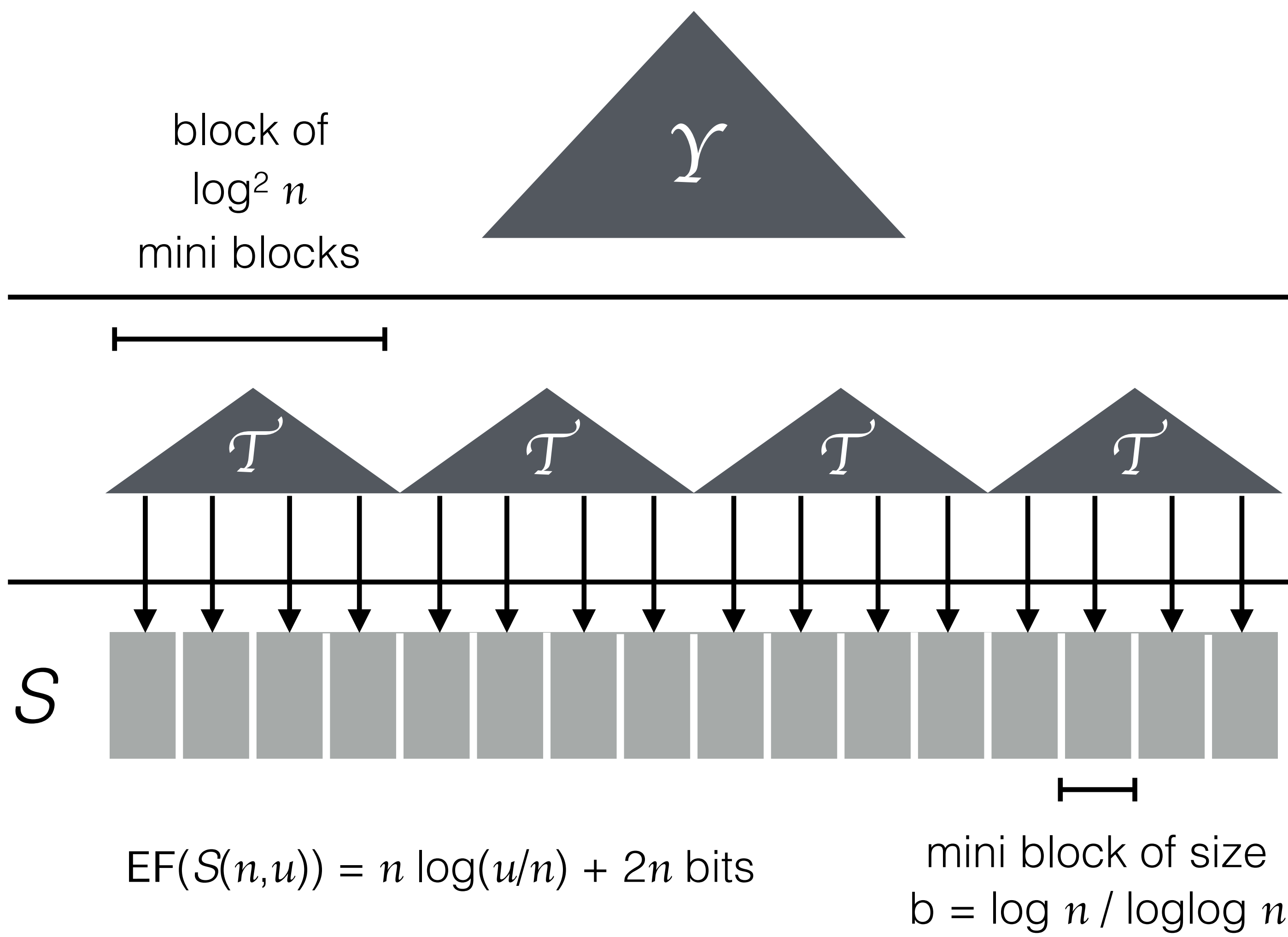
$\mathcal{T}$ is a ***k*-ary tree of constant height**:

- O(loglog n) time
- O($\log^2 n$ loglog n) bits

lower level

o(n) bits

Dynamic Elias-Fano Representation - CPM'17



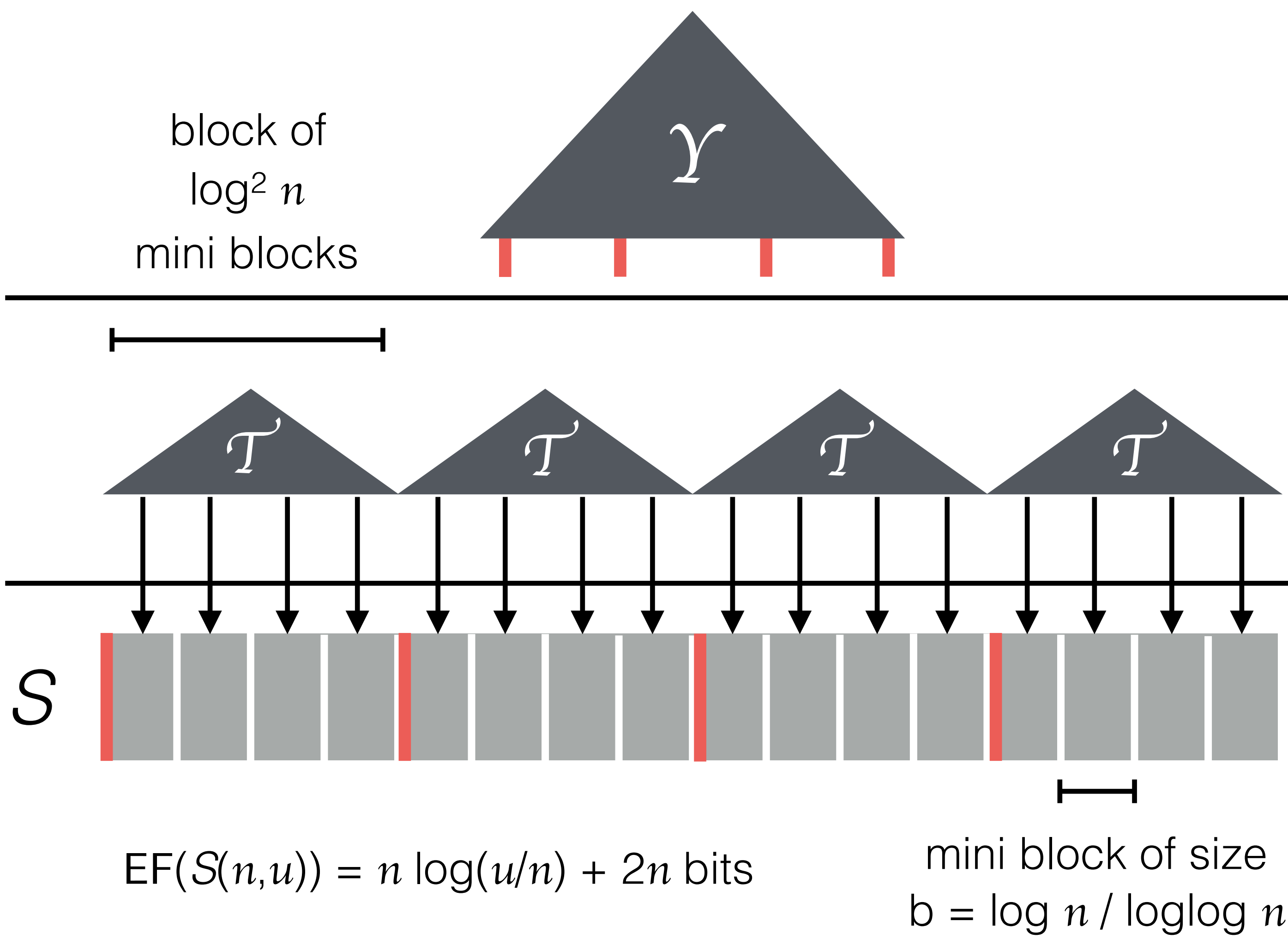
$\mathcal{T}$ is a ***k*-ary tree of constant height**:

- $O(\log\log n)$ time
- $O(\log^2 n \log\log n)$ bits

lower level

$o(n)$ bits

Dynamic Elias-Fano Representation - CPM'17



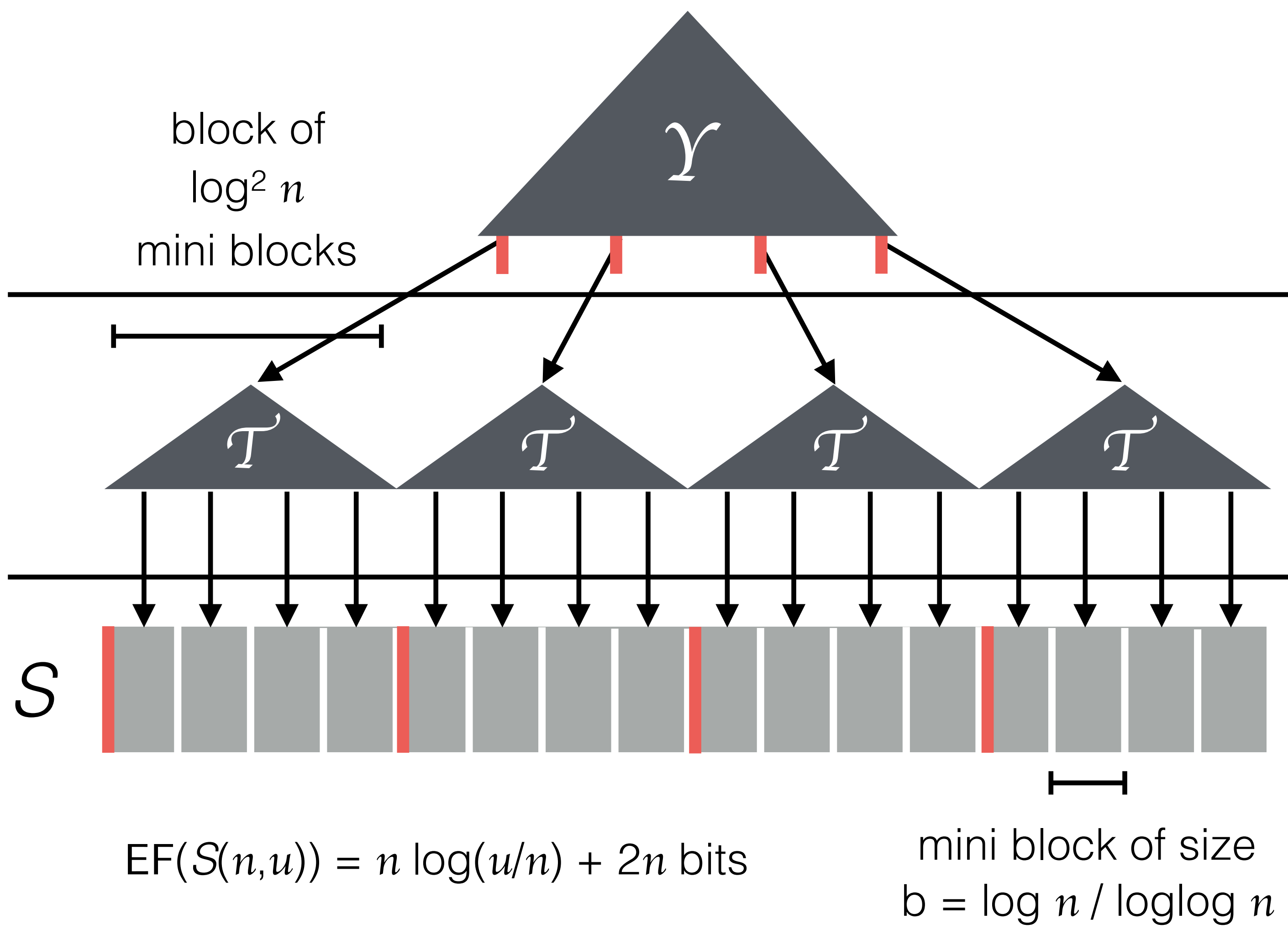
$\mathcal{T}$ is a ***k*-ary tree of constant height**:

- $O(\log\log n)$ time
- $O(\log^2 n \log\log n)$ bits

lower level

$o(n)$ bits

Dynamic Elias-Fano Representation - CPM'17



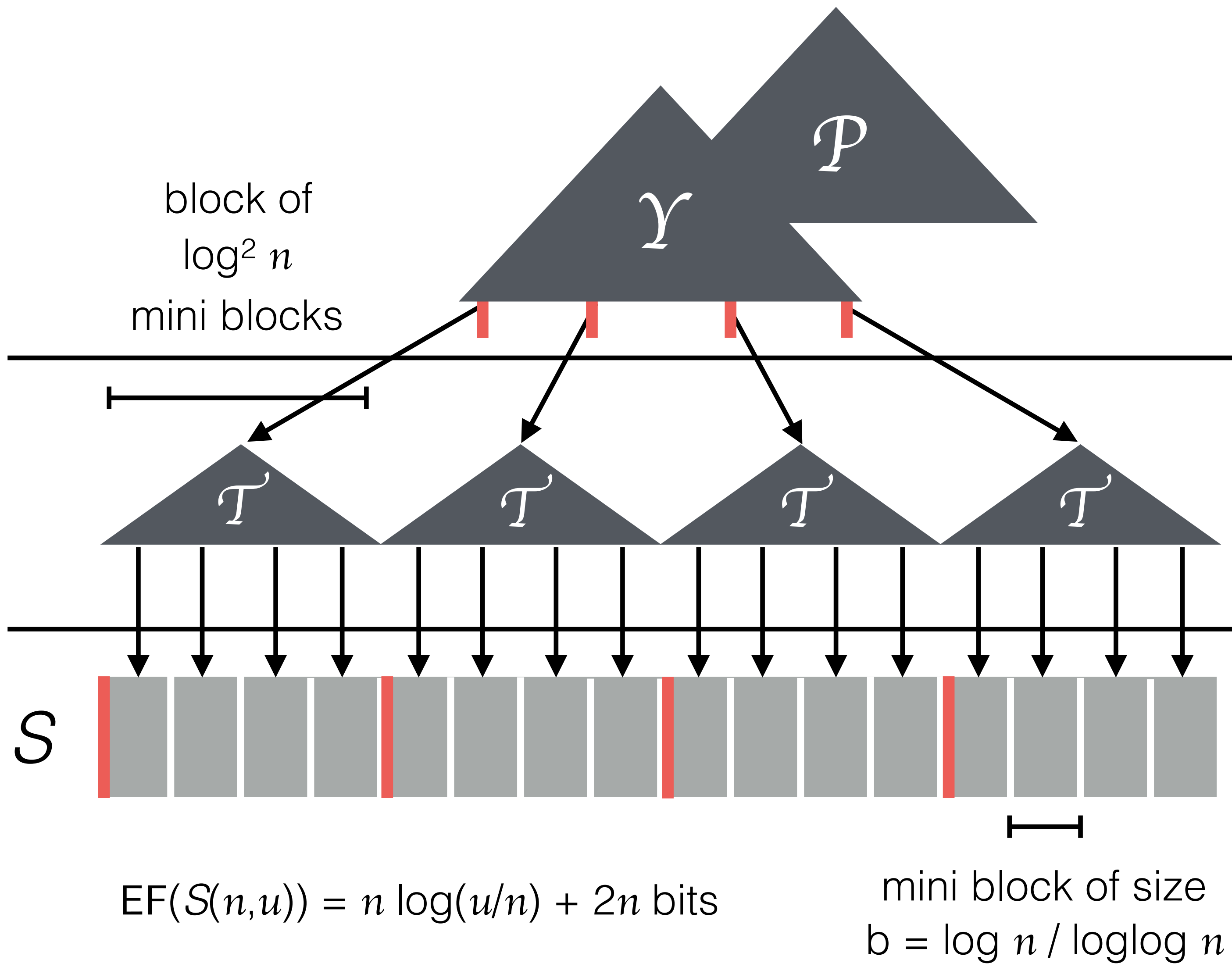
$\mathcal{T}$ is a ***k*-ary tree of constant height**:

- O(loglog n) time
- O($\log^2 n$ loglog n) bits

lower level

o(n) bits

Dynamic Elias-Fano Representation - CPM’17



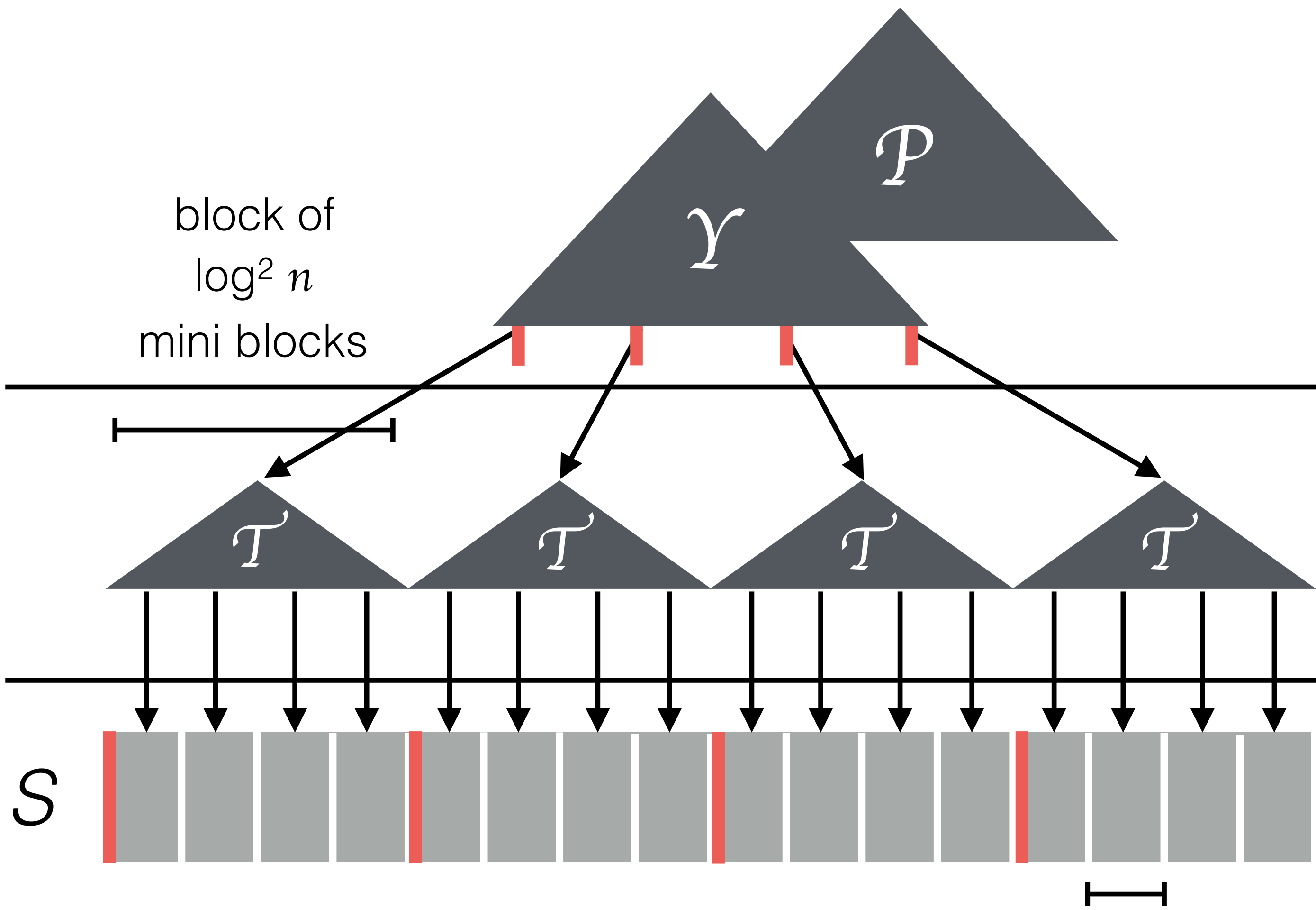
$\mathcal{T}$ is a ***k*-ary tree of constant height**:

- O($\log\log n$) time
- O($\log^2 n \log\log n$) bits

lower level

o(n) bits

Dynamic Elias-Fano Representation - CPM'17



$\mathcal{Y}$ is an **Y-fast trie**

- O(loglog n) time

$\mathcal{P}$ is a **dynamic prefix-sums** DS

- O(b) time

O(n / (b x $\log^2 n$)) x O(log u) = o(n) bits each

upper level

$\mathcal{T}$ is a ***k*-ary tree of constant height**:

- O(loglog n) time
- O($\log^2 n$ loglog n) bits

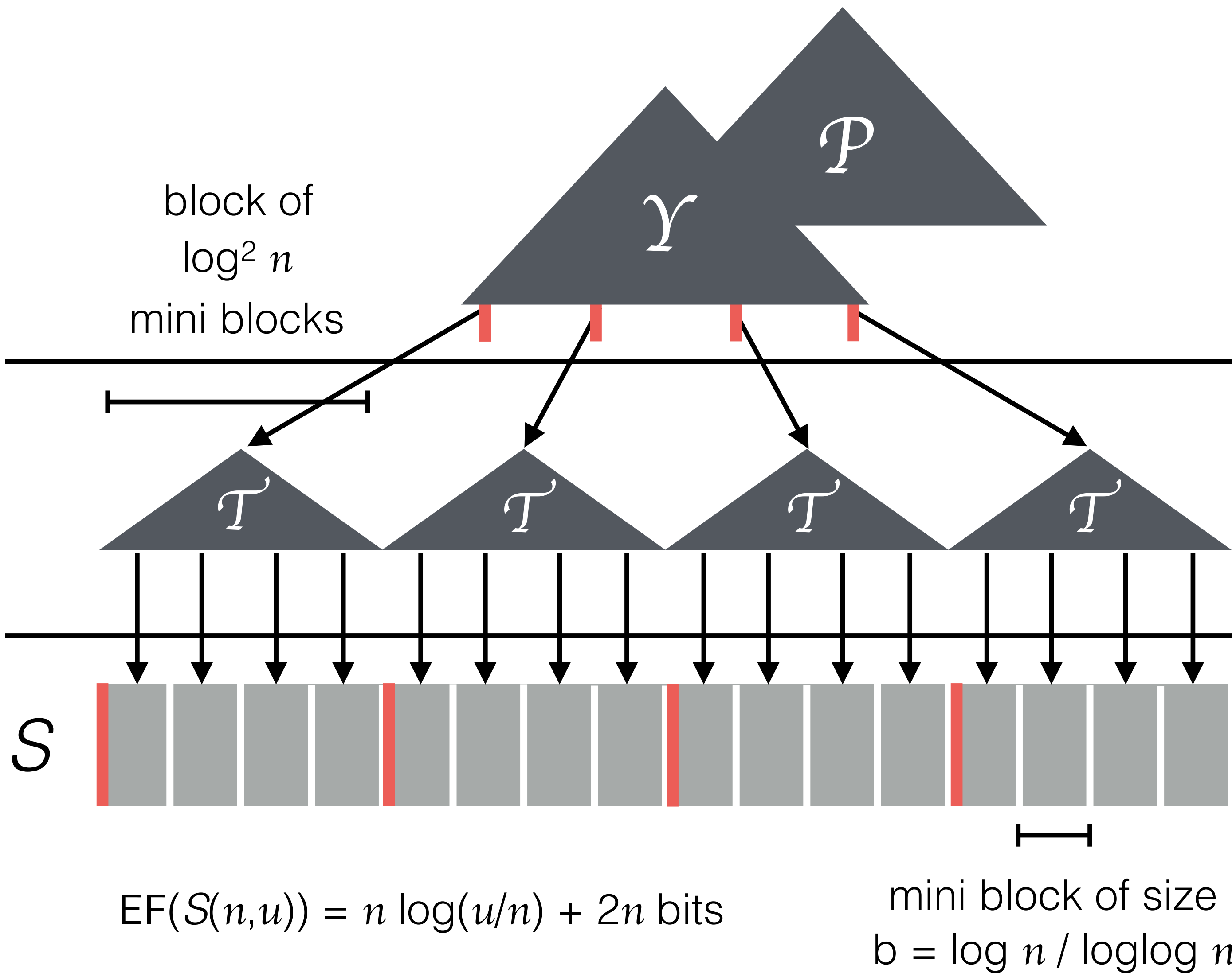
lower level

o(n) bits

EF($S(n,u)$) = n log(u/n) + 2n bits

mini block of size
b = log n / loglog n

Dynamic Elias-Fano Representation - CPM’17



Υ is an **Y-fast trie**

- O(loglog n) time

$\mathcal{P}$ is a **dynamic prefix-sums** DS

- O(b) time

O(n / (b x $\log^2 n$)) x O(log u) = o(n) bits each

upper level

o(n) bits

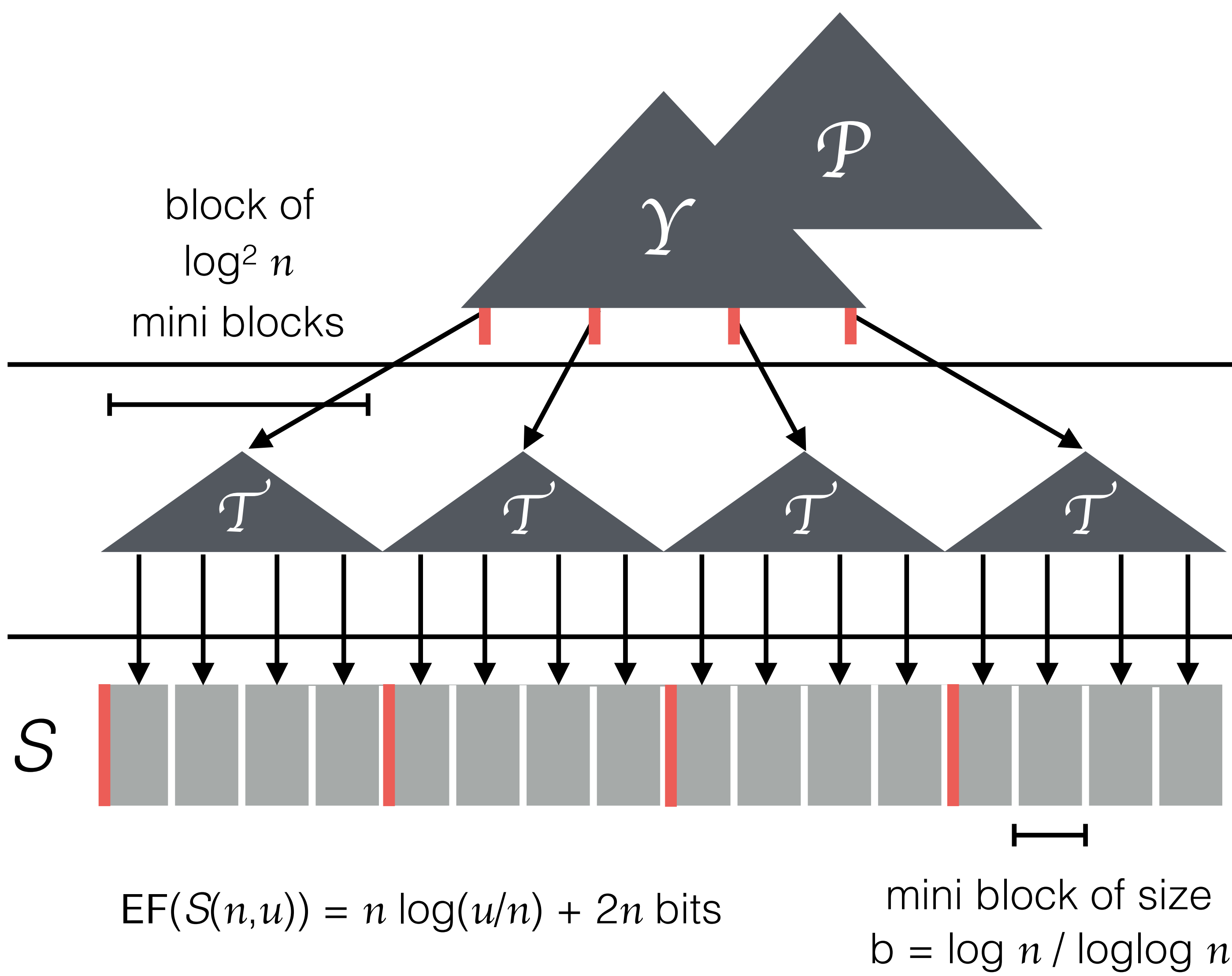
$\mathcal{T}$ is a ***k*-ary tree of constant height**:

- O(loglog n) time
- O($\log^2 n$ loglog n) bits

lower level

o(n) bits

Dynamic Elias-Fano Representation - CPM'17



$\mathcal{Y}$ is an **Y-fast trie**

- O(loglog n) time

$\mathcal{P}$ is a **dynamic prefix-sums** DS

- O(b) time

$O(n / (b \times \log^2 n)) \times O(\log u) = o(n)$ bits each

upper level

$o(n)$ bits

$\mathcal{T}$ is a ***k*-ary tree of constant height**:

- O(loglog n) time
- O($\log^2 n$ loglog n) bits

lower level

$o(n)$ bits

+ some technicalities

N-grams

Strings of *N* words.

N typically ranges from 1 to 5.

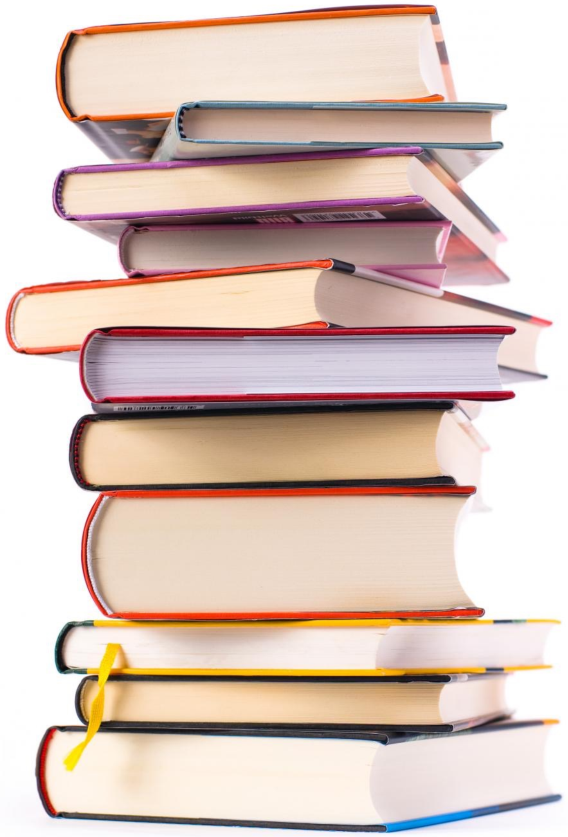
Extracted from text using a *sliding window* approach.

N-grams

Strings of *N* words.

N typically ranges from 1 to 5.

Extracted from text using a *sliding window* approach.

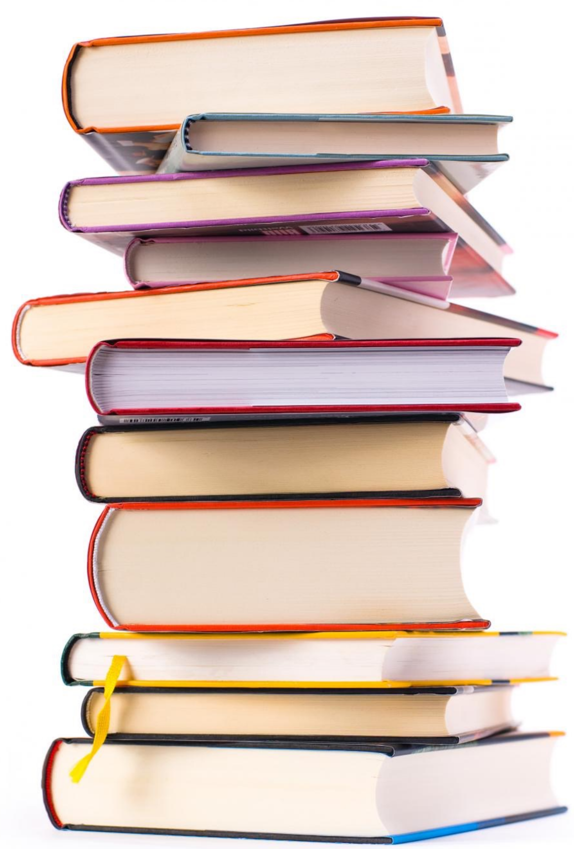


N-grams

Strings of *N* words.

N typically ranges from 1 to 5.

Extracted from text using a *sliding window* approach.



Google Books

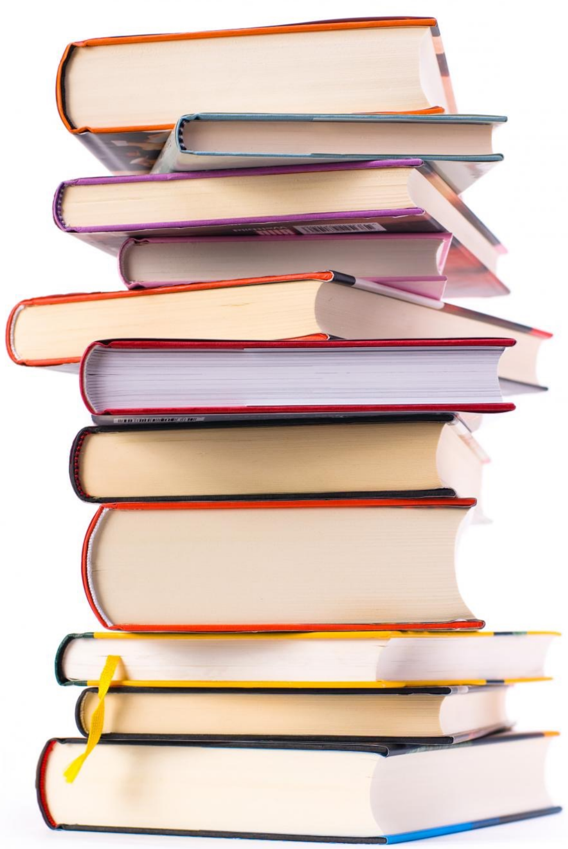
≈ 6% of the books ever published

N-grams

Strings of *N* words.

N typically ranges from 1 to 5.

Extracted from text using a *sliding window* approach.



Google Books

≈ 6% of the books ever published

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

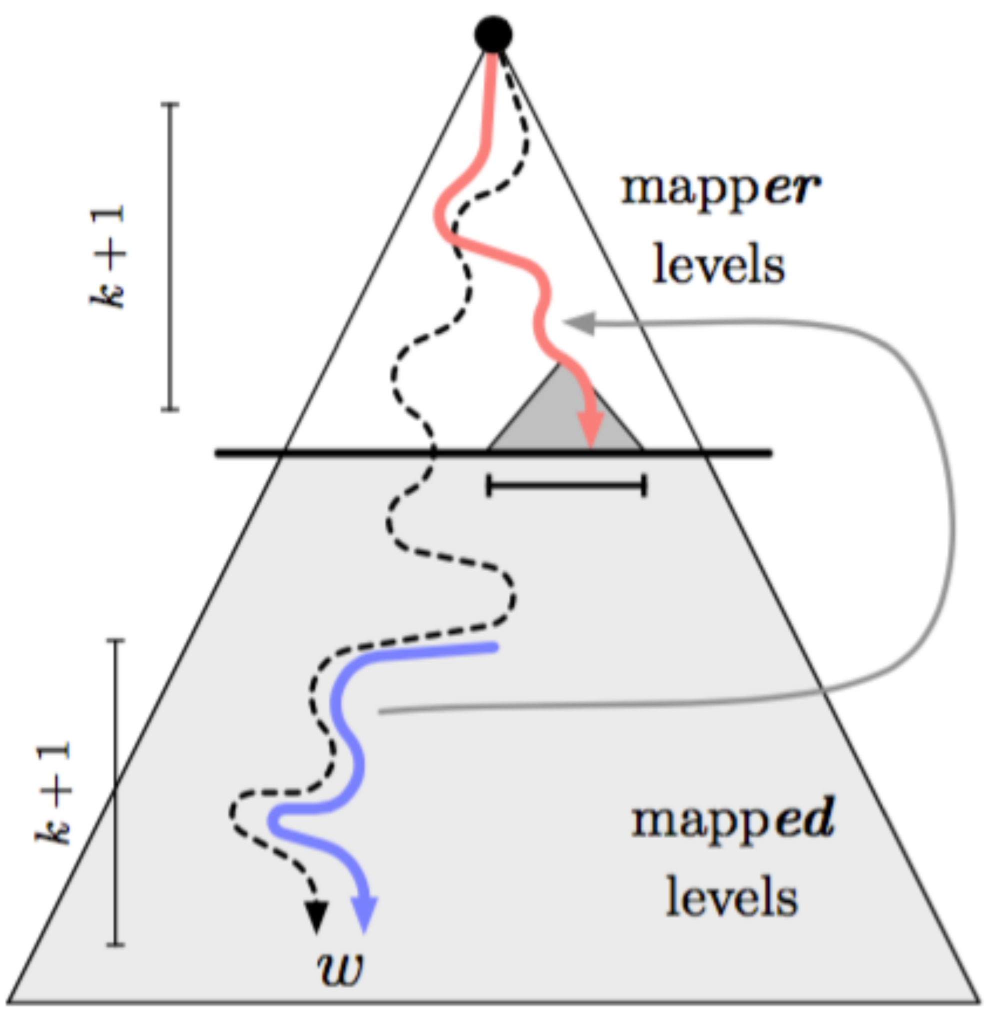
More than 11 billion grams.

Observation: the number of words following a given context is **small**.

High-level idea: map a word ID to the **position** it takes within its *sibling* IDs (the IDs following a context of fixed length k).

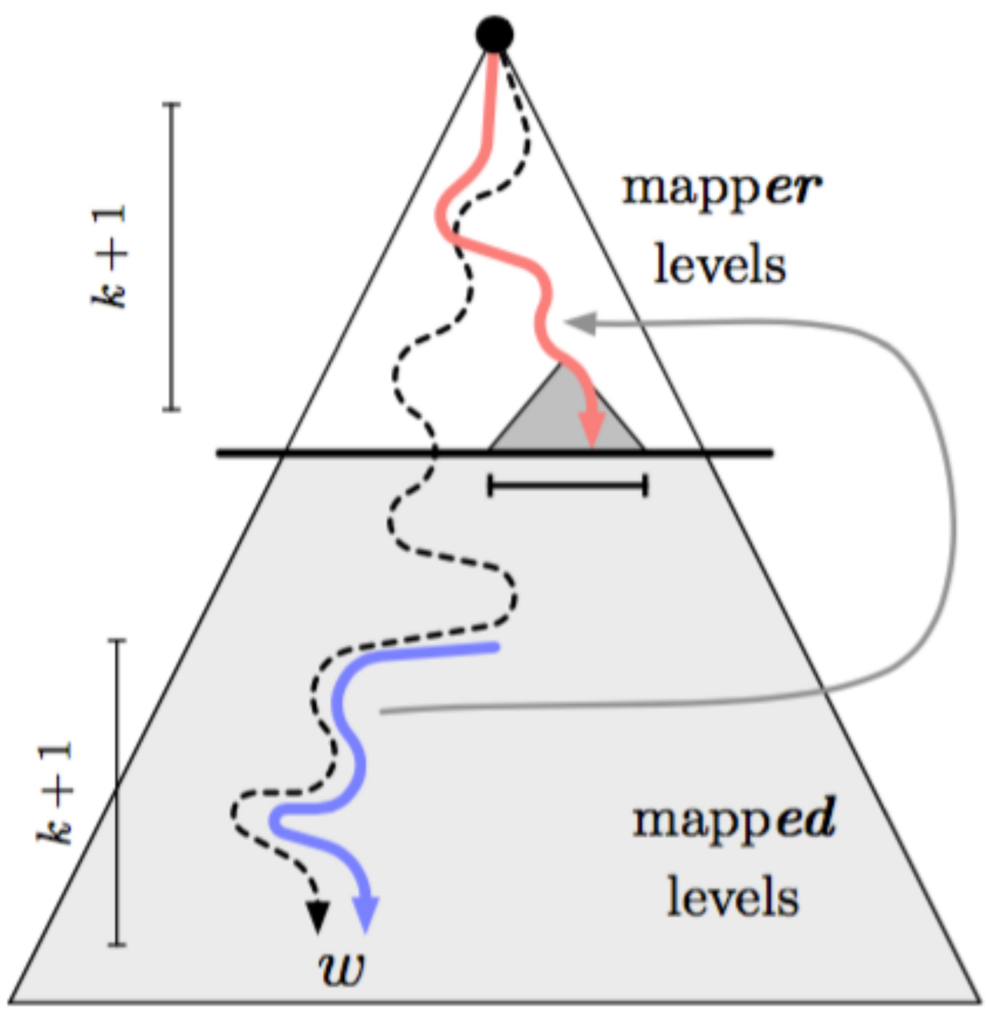
Observation: the number of words following a given context is **small**.

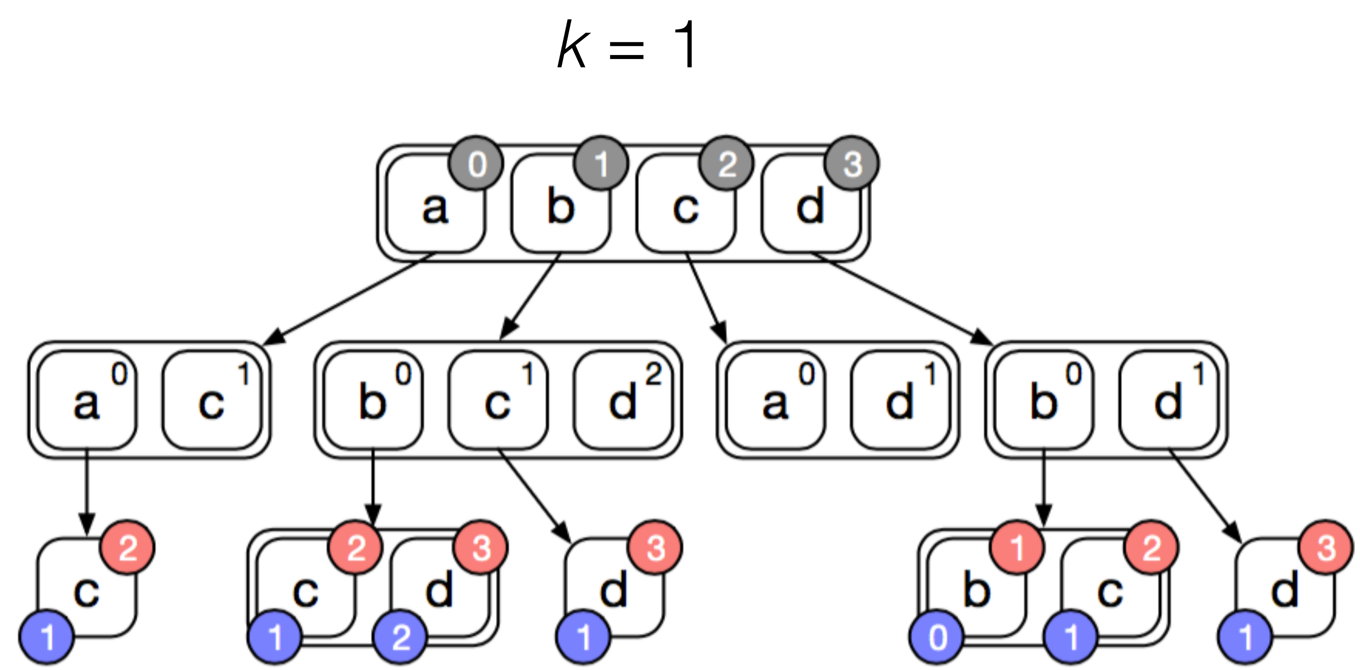
High-level idea: map a word ID to the **position** it takes within its *sibling* IDs (the IDs following a context of fixed length *k*).



Observation: the number of words following a given context is **small**.

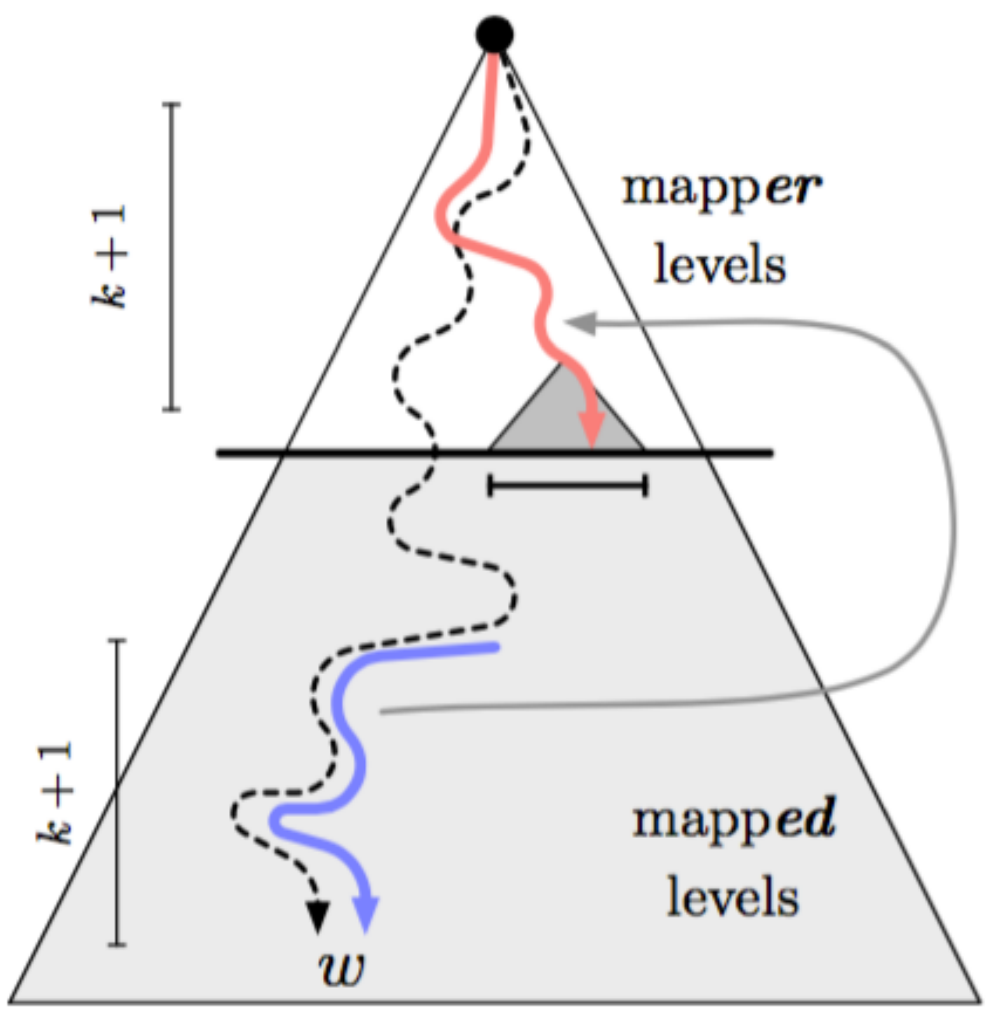
High-level idea: map a word ID to the **position** it takes within its *sibling* IDs (the IDs following a context of fixed length *k*).

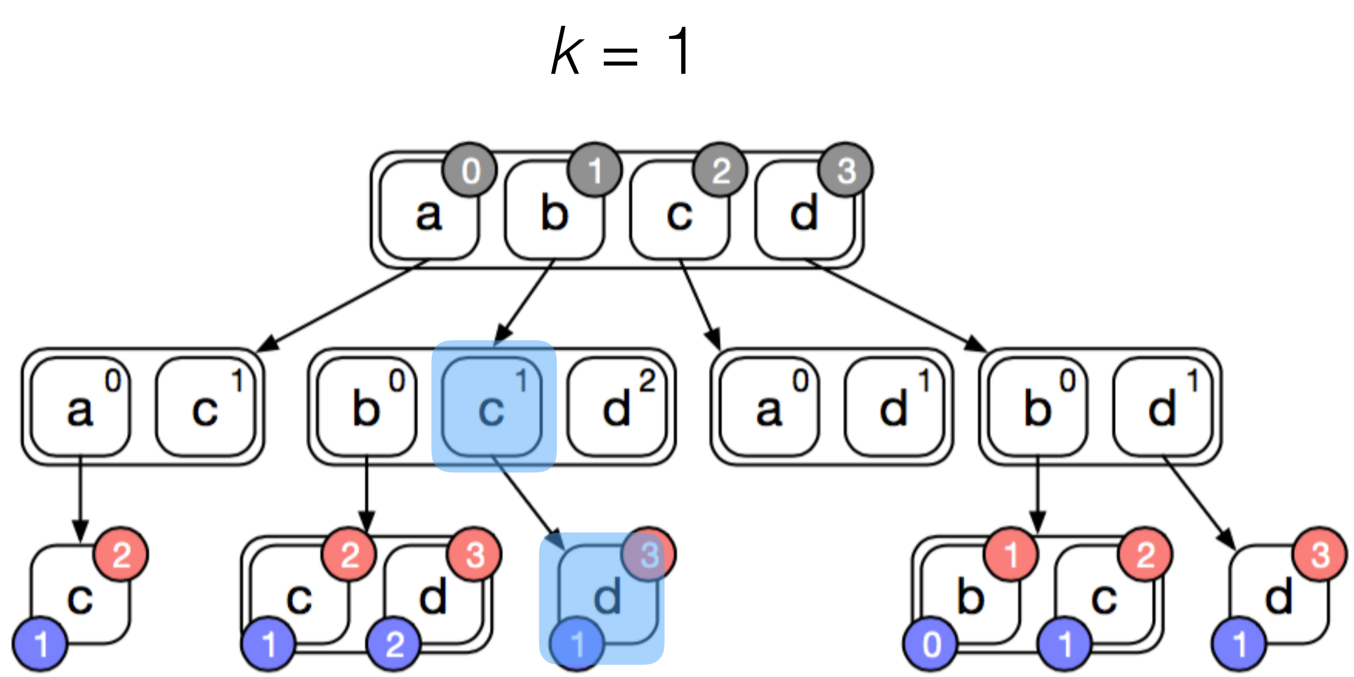




Observation: the number of words following a given context is **small**.

High-level idea: map a word ID to the **position** it takes within its *sibling* IDs (the IDs following a context of fixed length *k*).

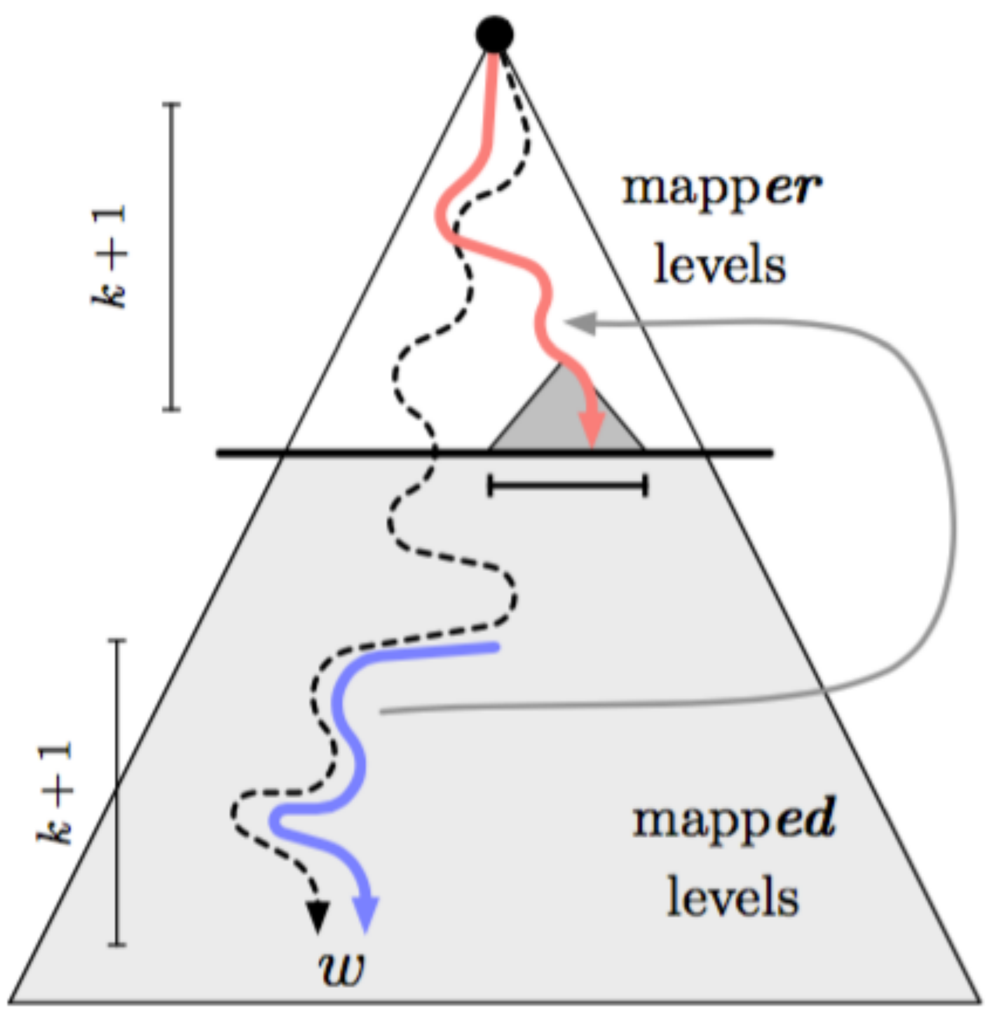


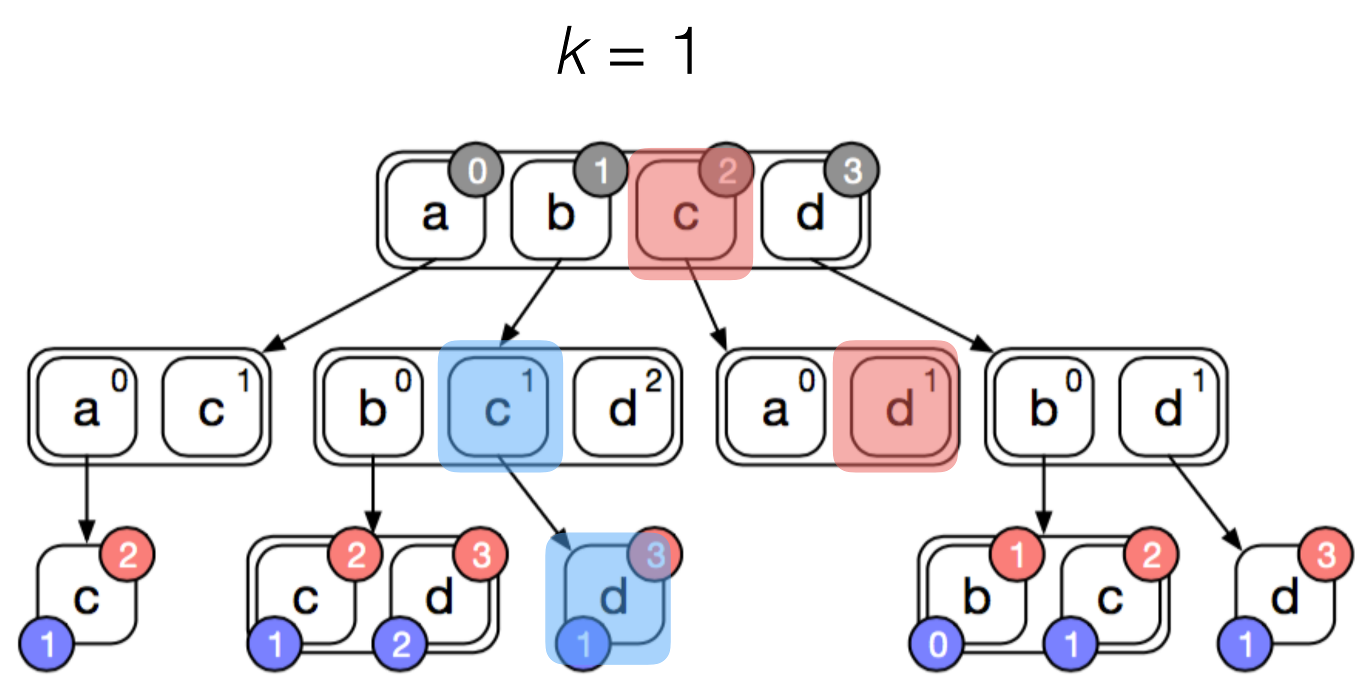


Compressed Tries with Context-based ID Remapping - SIGIR'17

Observation: the number of words following a given context is **small**.

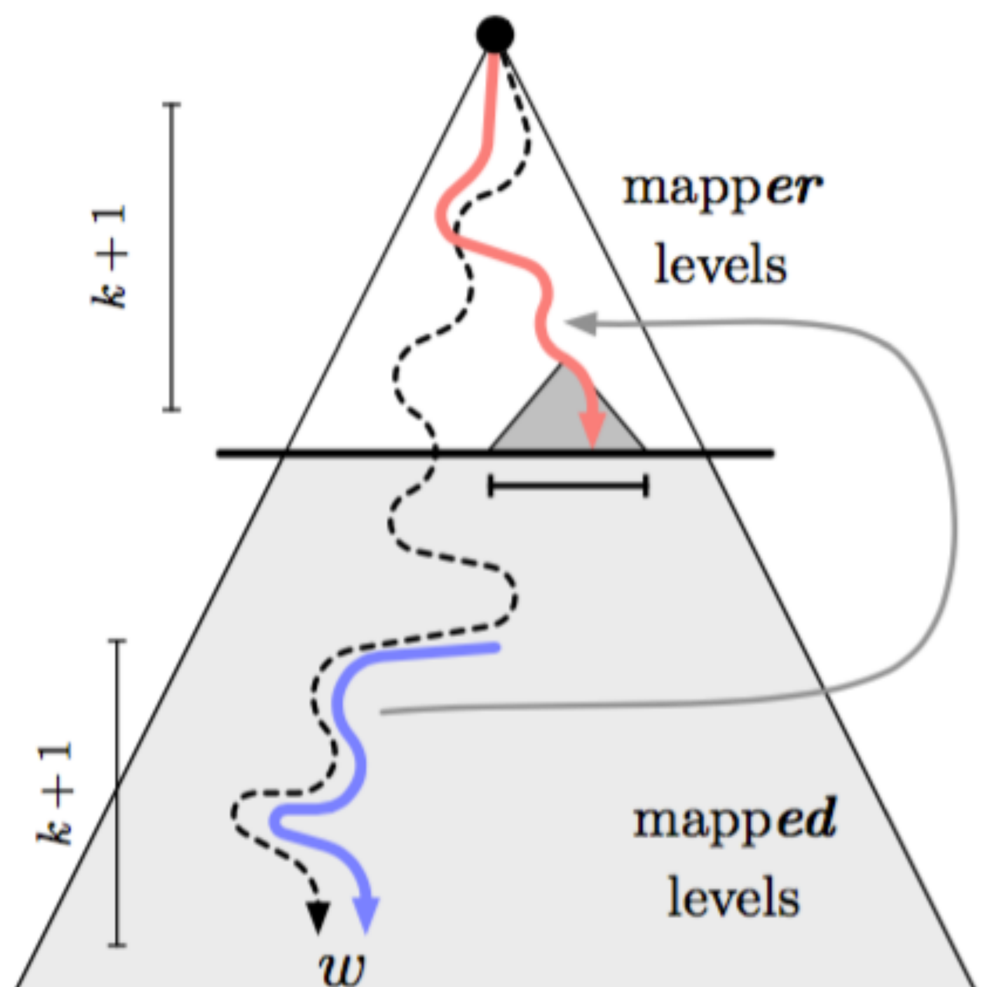
High-level idea: map a word ID to the **position** it takes within its *sibling* IDs (the IDs following a context of fixed length *k*).



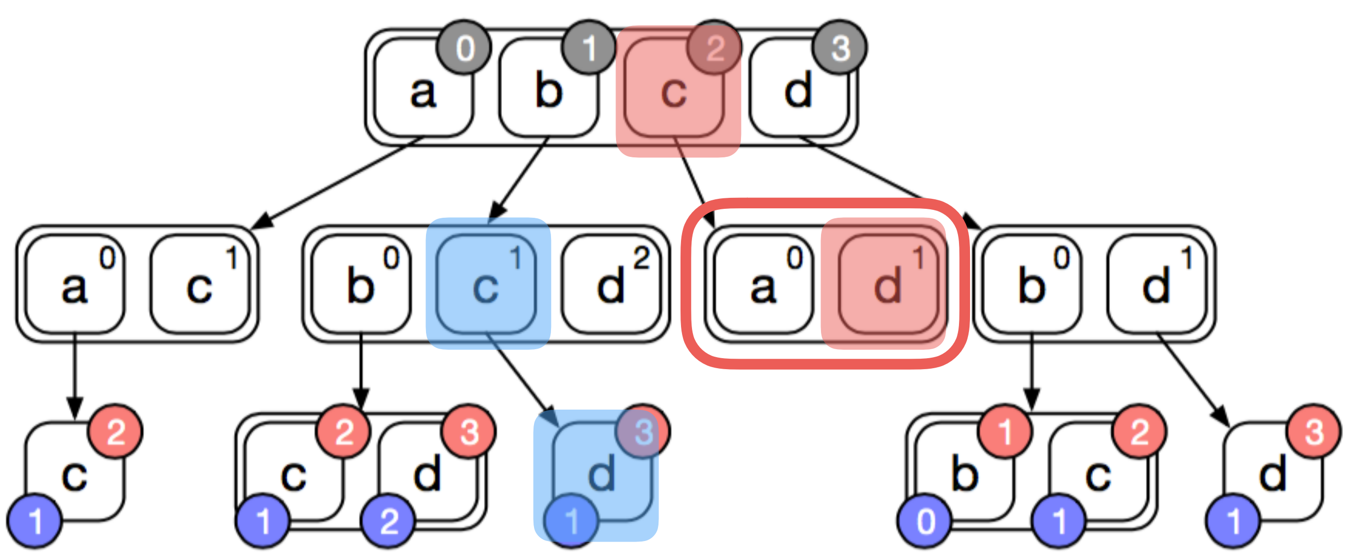


Observation: the number of words following a given context is **small**.

High-level idea: map a word ID to the **position** it takes within its *sibling* IDs (the IDs following a context of fixed length *k*).



k = 1



Observation: the number of words following a given context is **small**.

High-level idea: map a word ID to the **position** it takes within its *sibling* IDs (the IDs following a context of fixed length k).

Observation: the number of words following a given context is **small**.

High-level idea: map a word ID to the **position** it takes within its *sibling* IDs (the IDs following a context of fixed length *k*).

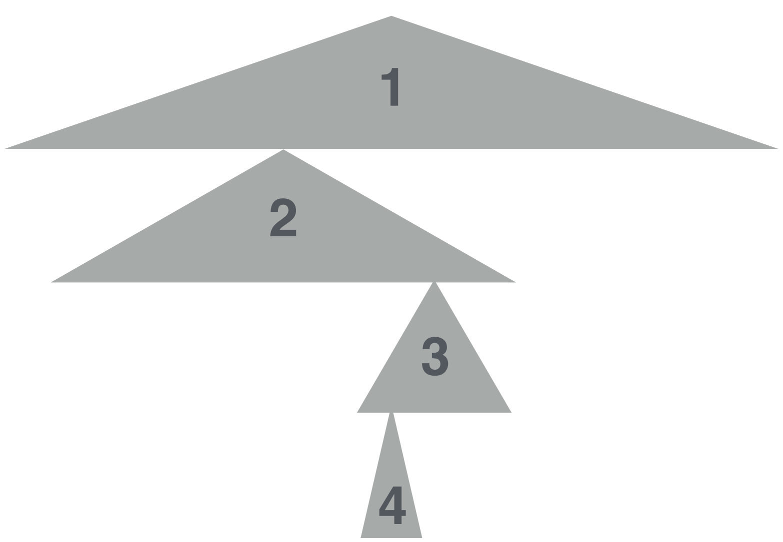
- **Millions** of unigrams.
- Height 5: **longer** contexts.
- The number of siblings has a **funnel-**shaped distribution.

Compressed Tries with Context-based ID Remapping - SIGIR'17

Observation: the number of words following a given context is **small**.

High-level idea: map a word ID to the **position** it takes within its *sibling* IDs (the IDs following a context of fixed length *k*).

- **Millions** of unigrams.
- Height 5: **longer** contexts.
- The number of siblings has a **funnel-**shaped distribution.

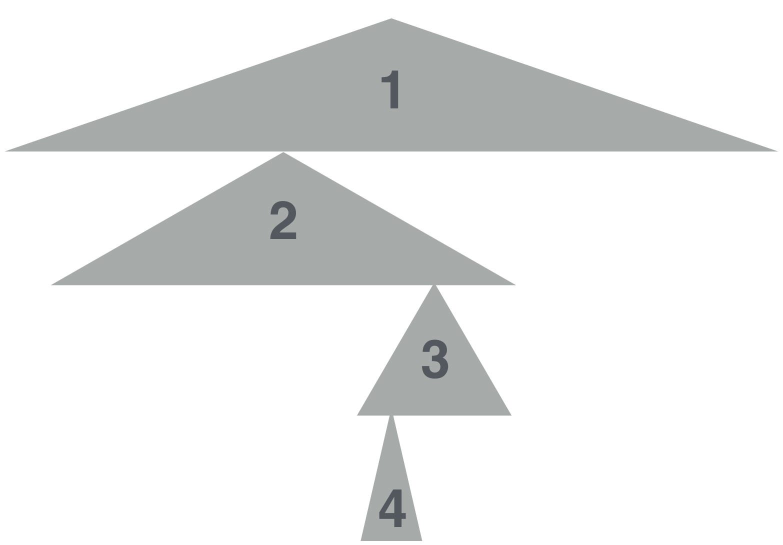


Compressed Tries with Context-based ID Remapping - SIGIR'17

Observation: the number of words following a given context is **small**.

High-level idea: map a word ID to the **position** it takes within its *sibling* IDs (the IDs following a context of fixed length *k*).

- **Millions** of unigrams.
- Height 5: **longer** contexts.
- The number of siblings has a **funnel-**shaped distribution.



u/n by varying context-length *k*

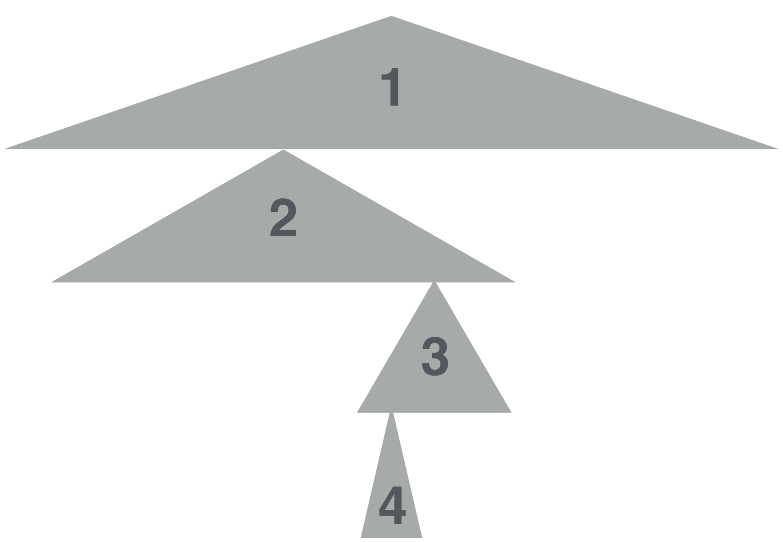
	k	3-grams	4-grams	5-grams
Europarl	0	2404	2782	2920
	1	213 (×11.28)	480 (×5.79)	646 (×4.52)
	2	2404	48 (×57.95)	101 (×28.91)
YahooV2	0	7350	7197	7417
	1	753 (×9.76)	1461 (×4.93)	1963 (×3.78)
	2	7350	104 (×69.20)	249 (×29.79)
GoogleV2	0	4050	6631	6793
	1	1025 (×3.95)	2192 (×3.03)	2772 (×2.45)
	2	4050	221 (×30.00)	503 (×13.50)

Compressed Tries with Context-based ID Remapping - SIGIR'17

Observation: the number of words following a given context is **small**.

High-level idea: map a word ID to the **position** it takes within its *sibling* IDs (the IDs following a context of fixed length *k*).

- **Millions** of unigrams.
- Height 5: **longer** contexts.
- The number of siblings has a **funnel-**shaped distribution.



u/n by varying context-length *k*

	k	3-grams		4-grams		5-grams	
Europarl	0	2404		2782		2920	
	1	213	(×11.28)	480	(×5.79)	646	(×4.52)
	2	2404		48	(×57.95)	101	(×28.91)
YahooV2	0	7350		7197		7417	
	1	753	(×9.76)	1461	(×4.93)	1963	(×3.78)
	2	7350		104	(×69.20)	249	(×29.79)
GoogleV2	0	4050		6631		6793	
	1	1025	(×3.95)	2192	(×3.03)	2772	(×2.45)
	2	4050		221	(×30.00)	503	(×13.50)

Compressed Tries with Context-based ID Remapping - SIGIR'17

N	Europarl n	YahooV2 n	GoogleV2 n
1	304 579	3 475 482	24 357 349
2	5 192 260	53 844 927	665 752 080
3	18 908 249	187 639 522	7 384 478 110
4	33 862 651	287 562 409	1 642 783 634
5	43 160 518	295 701 337	1 413 870 914
Total	101 428 257	828 223 677	11 131 242 087
gzip bpg	6.98	6.45	6.20

Test machine
Intel Xeon E5-2630 v3, 2.4 GHz
193 GB of RAM, Linux 64 bits

C++ implementation
`gcc` 5.4.1 with the highest optimization setting

Compressed Tries with Context-based ID Remapping - SIGIR'17

N	Europarl n	YahooV2 n	GoogleV2 n
1	304 579	3 475 482	24 357 349
2	5 192 260	53 844 927	665 752 080
3	18 908 249	187 639 522	7 384 478 110
4	33 862 651	287 562 409	1 642 783 634
5	43 160 518	295 701 337	1 413 870 914
Total	101 428 257	828 223 677	11 131 242 087
gzip bpg	6.98	6.45	6.20

Test machine
Intel Xeon E5-2630 v3, 2.4 GHz
193 GB of RAM, Linux 64 bits

C++ implementation
gcc 5.4.1 with the highest optimization setting

Compressed Tries with Context-based ID Remapping - SIGIR'17

N	Europarl n	YahooV2 n	GoogleV2 n
1	304 579	3 475 482	24 357 349
2	5 192 260	53 844 927	665 752 080
3	18 908 249	187 639 522	7 384 478 110
4	33 862 651	287 562 409	1 642 783 634
5	43 160 518	295 701 337	1 413 870 914
Total	101 428 257	828 223 677	11 131 242 087
gzip bpg	6.98	6.45	6.20

Test machine
Intel Xeon E5-2630 v3, 2.4 GHz
193 GB of RAM, Linux 64 bits

C++ implementation
gcc 5.4.1 with the highest
optimization setting

		Europarl		YahooV2		GoogleV2	
		bpg	$\mu s \times$ query	bpg	$\mu s \times$ query	bpg	$\mu s \times$ query
	EF	1.97	1.28	2.17	1.60	2.13	2.09
	PEF	1.87 (−4.99%)	1.35 (+5.93%)	1.91 (−12.03%)	1.73 (+8.00%)	1.52 (−28.60%)	1.91 (−8.79%)
CONTEXT-BASED ID REMAPPING $k = 1$	EF	1.67 (−15.30%)	1.58 (+23.86%)	1.89 (−12.92%)	2.05 (+28.07%)	1.91 (−10.24%)	3.03 (+44.61%)
	PEF	1.53 (−22.36%)	1.61 (+25.89%)	1.63 (−24.91%)	2.16 (+35.22%)	1.31 (−38.71%)	2.30 (+9.88%)
CONTEXT-BASED ID REMAPPING $k = 2$	EF	1.46 (−25.62%)	1.60 (+25.17%)	1.68 (−22.32%)	2.08 (+30.23%)	—	—
	PEF	1.28 (−34.87%)	1.64 (+28.12%)	1.38 (−36.15%)	2.15 (+34.81%)	—	—

Compressed Tries with Context-based ID Remapping - SIGIR'17

N	Europarl n	YahooV2 n	GoogleV2 n
1	304 579	3 475 482	24 357 349
2	5 192 260	53 844 927	665 752 080
3	18 908 249	187 639 522	7 384 478 110
4	33 862 651	287 562 409	1 642 783 634
5	43 160 518	295 701 337	1 413 870 914
Total	101 428 257	828 223 677	11 131 242 087
gzip bpg	6.98	6.45	6.20

Test machine
Intel Xeon E5-2630 v3, 2.4 GHz
193 GB of RAM, Linux 64 bits

C++ implementation
gcc 5.4.1 with the highest optimization setting

		Europarl		YahooV2		GoogleV2	
		bpg	$\mu s \times$ query	bpg	$\mu s \times$ query	bpg	$\mu s \times$ query
	EF	1.97	1.28	2.17	1.60	2.13	2.09
	PEF	1.87 (−4.99%)	1.35 (+5.93%)	1.91 (−12.03%)	1.73 (+8.00%)	1.52 (−28.60%)	1.91 (−8.79%)
CONTEXT-BASED ID REMAPPING $k = 1$	EF	1.67 (−15.30%)	1.58 (+23.86%)	1.89 (−12.92%)	2.05 (+28.07%)	1.91 (−10.24%)	3.03 (+44.61%)
CONTEXT-BASED ID REMAPPING $k = 1$	PEF	1.53 (−22.36%)	1.61 (+25.89%)	1.63 (−24.91%)	2.16 (+35.22%)	1.31 (−38.71%)	2.30 (+9.88%)
CONTEXT-BASED ID REMAPPING $k = 2$	EF	1.46 (−25.62%)	1.60 (+25.17%)	1.68 (−22.32%)	2.08 (+30.23%)	—	—
CONTEXT-BASED ID REMAPPING $k = 2$	PEF	1.28 (−34.87%)	1.64 (+28.12%)	1.38 (−36.15%)	2.15 (+34.81%)	—	—

Compressed Tries with Context-based ID Remapping - SIGIR’17

N	Europarl n	YahooV2 n	GoogleV2 n
1	304 579	3 475 482	24 357 349
2	5 192 260	53 844 927	665 752 080
3	18 908 249	187 639 522	7 384 478 110
4	33 862 651	287 562 409	1 642 783 634
5	43 160 518	295 701 337	1 413 870 914
Total	101 428 257	828 223 677	11 131 242 087
gzip bpg	6.98	6.45	6.20

Test machine
Intel Xeon E5-2630 v3, 2.4 GHz
193 GB of RAM, Linux 64 bits

C++ implementation
gcc 5.4.1 with the highest optimization setting

		Europarl		YahooV2		GoogleV2	
		bpg	$\mu s \times$ query	bpg	$\mu s \times$ query	bpg	$\mu s \times$ query
	EF	1.97	1.28	2.17	1.60	2.13	2.09
	PEF	1.87 (−4.99%)	1.35 (+5.93%)	1.91 (−12.03%)	1.73 (+8.00%)	1.52 (−28.60%)	1.91 (−8.79%)
CONTEXT-BASED ID REMAPPING $k = 1$	EF	1.67 (−15.30%)	1.58 (+23.86%)	1.89 (−12.92%)	2.05 (+28.07%)	1.91 (−10.24%)	3.03 (+44.61%)
CONTEXT-BASED ID REMAPPING $k = 1$	PEF	1.53 (−22.36%)	1.61 (+25.89%)	1.63 (−24.91%)	2.16 (+35.22%)	1.31 (−38.71%)	2.30 (+9.88%)
CONTEXT-BASED ID REMAPPING $k = 2$	EF	1.46 (−25.62%)	1.60 (+25.17%)	1.68 (−22.32%)	2.08 (+30.23%)	—	—
CONTEXT-BASED ID REMAPPING $k = 2$	PEF	1.28 (−34.87%)	1.64 (+28.12%)	1.38 (−36.15%)	2.15 (+34.81%)	—	—

Context-based ID Remapping

- reduces space by more than **36%** on average → **you will** notice this**!**

Compressed Tries with Context-based ID Remapping - SIGIR'17

N	Europarl n	YahooV2 n	GoogleV2 n
1	304 579	3 475 482	24 357 349
2	5 192 260	53 844 927	665 752 080
3	18 908 249	187 639 522	7 384 478 110
4	33 862 651	287 562 409	1 642 783 634
5	43 160 518	295 701 337	1 413 870 914
Total	101 428 257	828 223 677	11 131 242 087
gzip bpg	6.98	6.45	6.20

Test machine
Intel Xeon E5-2630 v3, 2.4 GHz
193 GB of RAM, Linux 64 bits

C++ implementation
gcc 5.4.1 with the highest optimization setting

		Europarl		YahooV2		GoogleV2	
		bpg	$\mu s \times$ query	bpg	$\mu s \times$ query	bpg	$\mu s \times$ query
	EF	1.97	1.28	2.17	1.60	2.13	2.09
	PEF	1.87 (−4.99%)	1.35 (+5.93%)	1.91 (−12.03%)	1.73 (+8.00%)	1.52 (−28.60%)	1.91 (−8.79%)
CONTEXT-BASED ID REMAPPING $k = 1$	EF	1.67 (−15.30%)	1.58 (+23.86%)	1.89 (−12.92%)	2.05 (+28.07%)	1.91 (−10.24%)	3.03 (+44.61%)
	PEF	1.53 (−22.36%)	1.61 (+25.89%)	1.63 (−24.91%)	2.16 (+35.22%)	1.31 (−38.71%)	2.30 (+9.88%)
CONTEXT-BASED ID REMAPPING $k = 2$	EF	1.46 (−25.62%)	1.60 (+25.17%)	1.68 (−22.32%)	2.08 (+30.23%)	—	—
	PEF	1.28 (−34.87%)	1.64 (+28.12%)	1.38 (−36.15%)	2.15 (+34.81%)	—	—

Context-based ID Remapping

- reduces space by more than **36%** on average → **you will** notice this**!**

Compressed Tries with Context-based ID Remapping - SIGIR'17

N	Europarl	YahooV2	GoogleV2
	n	n	n
1	304 579	3 475 482	24 357 349
2	5 192 260	53 844 927	665 752 080
3	18 908 249	187 639 522	7 384 478 110
4	33 862 651	287 562 409	1 642 783 634
5	43 160 518	295 701 337	1 413 870 914
Total	101 428 257	828 223 677	11 131 242 087
gzip bpg	6.98	6.45	6.20

Test machine
Intel Xeon E5-2630 v3, 2.4 GHz
193 GB of RAM, Linux 64 bits

C++ implementation
gcc 5.4.1 with the highest optimization setting

		Europarl		YahooV2		GoogleV2	
		bpg	$\mu s \times$ query	bpg	$\mu s \times$ query	bpg	$\mu s \times$ query
	EF	1.97	1.28	2.17	1.60	2.13	2.09
	PEF	1.87 (−4.99%)	1.35 (+5.93%)	1.91 (−12.03%)	1.73 (+8.00%)	1.52 (−28.60%)	1.91 (−8.79%)
CONTEXT-BASED ID REMAPPING $k = 1$	EF	1.67 (−15.30%)	1.58 (+23.86%)	1.89 (−12.92%)	2.05 (+28.07%)	1.91 (−10.24%)	3.03 (+44.61%)
	PEF	1.53 (−22.36%)	1.61 (+25.89%)	1.63 (−24.91%)	2.16 (+35.22%)	1.31 (−38.71%)	2.30 (+9.88%)
CONTEXT-BASED ID REMAPPING $k = 2$	EF	1.46 (−25.62%)	1.60 (+25.17%)	1.68 (−22.32%)	2.08 (+30.23%)	—	—
	PEF	1.28 (−34.87%)	1.64 (+28.12%)	1.38 (−36.15%)	2.15 (+34.81%)	—	—

Context-based ID Remapping

- reduces space by more than **36%** on average → **you will** notice this**!**
- brings approximately **30%** more time → **will you** notice this**?**

Compressed Tries with Context-based ID Remapping - SIGIR'17

	Europarl		YahooV2		GoogleV2	
	bpg	$\mu s \times$ query	bpg	$\mu s \times$ query	bpg	$\mu s \times$ query
PEF-Trie	1.87	1.35	1.91	1.73	1.52	1.91
PEF-RTrie	1.28	1.64	1.38	2.15	1.31	2.30
BerkeleyLM C.	1.70 (−8.89%) (+32.90%)	2.83 (+108.88%) (+72.70%)	1.69 (−11.41%) (+22.04%)	3.48 (+101.84%) (+61.70%)	1.45 (−4.87%) (+10.83%)	4.13 (+116.57%) (+79.76%)
BerkeleyLM H.3	6.70 (+258.81%) (+423.40%)	0.97 (−28.46%) (−40.85%)	7.82 (+310.38%) (+465.36%)	1.13 (−34.35%) (−47.41%)	9.24 (+507.79%) (+608.07%)	2.18 (+13.95%) (−5.42%)
BerkeleyLM H.50	7.96 (+326.03%) (+521.45%)	0.97 (−28.49%) (−40.88%)	9.37 (+391.32%) (+576.87%)	0.96 (−44.27%) (−55.35%)	—	—
Expgram	2.06 (+10.18%) (+60.73%)	2.80 (+106.61%) (+70.82%)	2.24 (+17.36%) (+61.68%)	9.23 (+435.33%) (+328.87%)	—	—
KenLM T.	2.99 (+60.11%) (+133.56%)	1.28 (−5.47%) (−21.84%)	3.44 (+80.39%) (+148.52%)	1.94 (+12.32%) (−10.01%)	—	—
Marisa	3.61 (+93.09%) (+181.66%)	2.06 (+52.00%) (+25.67%)	3.81 (+99.60%) (+174.98%)	3.24 (+87.96%) (+50.58%)	—	—
RandLM	1.81 (−3.06%) (+41.41%)	4.39 (+224.20%) (+168.04%)	2.02 (+6.18%) (+46.29%)	5.08 (+194.35%) (+135.82%)	2.60 (+70.73%) (+98.90%)	9.25 (+384.54%) (+302.19%)

Compressed Tries with Context-based ID Remapping - SIGIR’17

	Europarl		YahooV2		GoogleV2	
	bpg	μs × query	bpg	μs × query	bpg	μs × query
PEF-Trie	1.87	1.35	1.91	1.73	1.52	1.91
PEF-RTrie	1.28	1.64	1.38	2.15	1.31	2.30
BerkeleyLM C.	1.70 (−8.89%) (+32.90%)	2.83 (+108.88%) (+72.70%)	1.69 (−11.41%) (+22.04%)	3.48 (+101.84%) (+61.70%)	1.45 (−4.87%) (+10.83%)	4.13 (+116.57%) (+79.76%)
BerkeleyLM H.3	6.70 (+258.81%) (+423.40%)	0.97 (−28.46%) (−40.85%)	7.82 (+310.38%) (+465.36%)	1.13 (−34.35%) (−47.41%)	9.24 (+507.79%) (+608.07%)	2.18 (+13.95%) (−5.42%)
BerkeleyLM H.50	7.96 (+326.03%) (+521.45%)	0.97 (−28.49%) (−40.88%)	9.37 (+391.32%) (+576.87%)	0.96 (−44.27%) (−55.35%)	—	—
Expgram	2.06 (+10.18%) (+60.73%)	2.80 (+106.61%) (+70.82%)	2.24 (+17.36%) (+61.68%)	9.23 (+435.33%) (+328.87%)	—	—
KenLM T.	2.99 (+60.11%) (+133.56%)	1.28 (−5.47%) (−21.84%)	3.44 (+80.39%) (+148.52%)	1.94 (+12.32%) (−10.01%)	—	—
Marisa	3.61 (+93.09%) (+181.66%)	2.06 (+52.00%) (+25.67%)	3.81 (+99.60%) (+174.98%)	3.24 (+87.96%) (+50.58%)	—	—
RandLM	1.81 (−3.06%) (+41.41%)	4.39 (+224.20%) (+168.04%)	2.02 (+6.18%) (+46.29%)	5.08 (+194.35%) (+135.82%)	2.60 (+70.73%) (+98.90%)	9.25 (+384.54%) (+302.19%)

	Europarl		YahooV2		GoogleV2	
	bpg	μs × query	bpg	μs × query	bpg	μs × query
PEF-Trie	1.87	1.35	1.91	1.73	1.52	1.91
PEF-RTrie	1.28	1.64	1.38	2.15	1.31	2.30
BerkeleyLM C.	1.70 (−8.89%) (+32.90%)	2.83 (+108.88%) (+72.70%)	1.69 (−11.41%) (+22.04%)	3.48 (+101.84%) (+61.70%)	1.45 (−4.87%) (+10.83%)	4.13 (+116.57%) (+79.76%)
BerkeleyLM H.3	6.70 (+258.81%) (+423.40%)	0.97 (−28.46%) (−40.85%)	7.82 (+310.38%) (+465.36%)	1.13 (−34.35%) (−47.41%)	9.24 (+507.79%) (+608.07%)	2.18 (+13.95%) (−5.42%)
BerkeleyLM H.50	7.96 (+326.03%) (+521.45%)	0.97 (−28.49%) (−40.88%)	9.37 (+391.32%) (+576.87%)	0.96 (−44.27%) (−55.35%)	—	—
Expgram	2.06 (+10.18%) (+60.73%)	2.80 (+106.61%) (+70.82%)	2.24 (+17.36%) (+61.68%)	9.23 (+435.33%) (+328.87%)	—	—
KenLM T.	2.99 [illegible] **2.3X** (+133.56%)	1.28 (−5.47%) (−21.84%)	3.44 [illegible] **2.5X** (+148.52%)	1.94 (+12.32%) (−10.01%)	—	—
Marisa	3.61 (+93.09%) (+181.66%)	2.06 (+52.00%) (+25.67%)	3.81 (+99.60%) (+174.98%)	3.24 (+87.96%) (+50.58%)	—	—
RandLM	1.81 (−3.06%) (+41.41%)	4.39 (+224.20%) (+168.04%)	2.02 (+6.18%) (+46.29%)	5.08 (+194.35%) (+135.82%)	2.60 (+70.73%) (+98.90%)	9.25 (+384.54%) (+302.19%)

Compressed Tries with Context-based ID Remapping - SIGIR'17

	Europarl bpg		Europarl µs × query		YahooV2 bpg		YahooV2 µs × query		GoogleV2 bpg		GoogleV2 µs × query	
PEF-Trie	1.87		1.35		1.91		1.73		1.52		1.91	
PEF-RTrie	1.28		1.64		1.38		2.15		1.31		2.30	
BerkeleyLM C.	1.70	(−8.89%) (+32.90%)	2.83	(+108.88%) (+72.70%)	1.69	(−11.41%) (+22.04%)	3.48	(+101.84%) (+61.70%)	1.45	(−4.87%) (+10.83%)	4.13	(+116.57%) (+79.76%)
BerkeleyLM H.3	6.70	(+258.81%) (+423.40%)	0.97	(−28.46%) (−40.85%)	7.82	(+310.38%) (+465.36%)	1.13	(−34.35%) (−47.41%)	9.24	(+507.79%) (+608.07%)	2.18	(+13.95%) (−5.42%)
BerkeleyLM H.50	7.96	(+326.03%) (+521.45%)	0.97	(−28.49%) (−40.88%)	9.37	(+391.32%) (+576.87%)	0.96	(−44.27%) (−55.35%)	—		—	
Expgram	2.06	(+10.18%) (+60.73%)	2.80	(+106.61%) (+70.82%)	2.24	(+17.36%) (+61.68%)	9.23	(+435.33%) (+328.87%)	—		—	
KenLM T.	2.99	[illegible] (+133.56%)	1.28	(−5.47%) (−21.84%)	3.44	[illegible] (+148.52%)	1.94	(+12.32%) (−10.01%)	—		—	
Marisa	3.61	(+93.09%) (+181.66%)	2.06	(+52.00%) (+25.67%)	3.81	(+99.60%) (+174.98%)	3.24	(+87.96%) (+50.58%)	—		—	
RandLM	1.81	(−3.06%) (+41.41%)	4.39	(+224.20%) (+168.04%)	2.02	(+6.18%) (+46.29%)	5.08	(+194.35%) (+135.82%)	2.60	(+70.73%) (+98.90%)	9.25	(+384.54%) (+302.19%)

2.3X

2.5X

Compressed Tries with Context-based ID Remapping - SIGIR'17

	Europarl		YahooV2		GoogleV2	
	bpg	$\mu s \times$ query	bpg	$\mu s \times$ query	bpg	$\mu s \times$ query
PEF-Trie	1.87	1.35	1.91	1.73	1.52	1.91
PEF-RTrie	1.28	1.64	1.38	2.15	1.31	2.30
BerkeleyLM C.	1.70 (−8.89%) (+32.90%)	2.83 ([illegible]) (+72.70%) 2X	1.69 (−11.41%) (+22.04%)	3.48 ([illegible]) (+61.70%) 2X	1.45 (−4.87%) (+10.83%)	4.13 ([illegible]) (+79.76%) 2X
BerkeleyLM H.3	6.70 (+258.81%) ([illegible]) 2.5÷	0.97 (−28.46%) (−40.85%)	7.82 (+310.38%) ([illegible]) 3.1÷	1.13 (−34.35%) (−47.41%)	9.24 ([illegible]) ([illegible]) 5.5X	2.18 (+13.95%) (−5.42%)
BerkeleyLM H.50	7.96 ([illegible]) (+521.45%) 5.2X	0.97 (−28.49%) (−40.88%)	9.37 ([illegible]) (+576.87%) 5.8X	0.96 (−44.27%) (−55.35%)	—	—
Expgram	2.06 (+10.18%) (+60.73%)	2.80 ([illegible]) (+70.82%) 2X	2.24 (+17.36%) (+61.68%)	9.23 ([illegible]) ([illegible]) 3.5X	—	—
KenLM T.	2.99 ([illegible]) (+133.56%) 2.3X	1.28 (−5.47%) (−21.84%)	3.44 ([illegible]) (+148.52%) 2.5X	1.94 (+12.32%) (−10.01%)	—	—
Marisa	3.61 (+93.09%) ([illegible]) 2.8X	2.06 (+52.00%) (+25.67%)	3.81 (+99.60%) ([illegible]) 2.7X	3.24 (+87.96%) (+50.58%)	—	—
RandLM	1.81 (−3.06%) (+41.41%)	4.39 ([illegible]) (+168.04%) 2.5X	2.02 (+6.18%) (+46.29%)	5.08 ([illegible]) (+135.82%) 2.5X	2.60 (+70.73%) (+98.90%)	9.25 ([illegible]) (+302.19%) 3X

Compressed Tries with Context-based ID Remapping - SIGIR'17

	Europarl		YahooV2		GoogleV2	
	bpg	$\mu s \times$ query	bpg	$\mu s \times$ query	bpg	$\mu s \times$ query
PEF-Trie	1.87	1.35	1.91	1.73	1.52	1.91
PEF-RTrie	1.28	1.64	1.38	2.15	1.31	2.30
BerkeleyLM C.	1.70 (−8.89%) (+32.90%)	2.83 (+[illegible]) (+72.70%) 2X	1.69 (−11.41%) (+22.04%)	3.48 (+[illegible]) (+61.70%) 2X	1.45 (−4.87%) (+10.83%)	4.13 (+[illegible]) (+79.76%) 2X
BerkeleyLM H.3	6.70 (+258.81%) (+[illegible]) 2.5÷	0.97 (−28.46%) (−40.85%)	7.82 (+310.38%) (+[illegible]) 3.1÷	1.13 (−34.35%) (−47.41%)	9.24 (+507.70%) (+608.07%) 5.5X	2.18 (+13.95%) (−5.42%)
BerkeleyLM H.50	7.96 (+[illegible]) (+521.45%) 5.2X	0.97 (−28.49%) (−40.88%)	9.37 (+[illegible]) (+576.87%) 5.8X	0.96 (−44.27%) (−55.35%)	—	—
Expgram	2.06 (+10.18%) (+60.73%)	2.80 (+[illegible]) (+70.82%) 2X	2.24 (+17.36%) (+61.68%)	9.23 (+[illegible]) (+[illegible]) 3.5X	—	—
KenLM T.	2.99 (+[illegible]) (+133.56%) 2.3X	1.28 (−5.47%) (−21.84%)	3.44 (+[illegible]) (+148.52%) 2.5X	1.94 (+12.32%) (−10.01%)	—	—
Marisa	3.61 (+93.09%) (+[illegible]) 2.8X	2.06 (+52.00%) (+25.67%)	3.81 (+99.60%) (+[illegible]) 2.7X	3.24 (+87.96%) (+50.58%)	—	—
RandLM	1.81 (−3.06%) (+41.41%)	4.39 (+[illegible]) (+168.04%) 2.5X	2.02 (+6.18%) (+46.29%)	5.08 (+[illegible]) (+135.82%) 2.5X	2.60 (+70.73%) (+98.90%)	9.25 (+[illegible]) (+302.19%) 3X

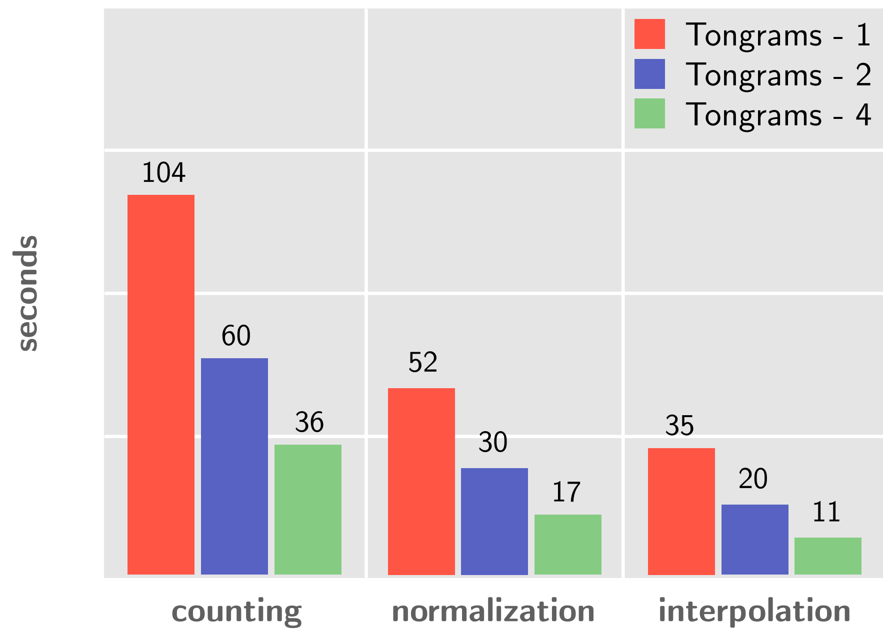
Compressed Tries with Context-based ID Remapping - SIGIR'17

	Europarl		YahooV2		GoogleV2	
	bpg	μs × query	bpg	μs × query	bpg	μs × query
PEF-Trie	1.87	1.35	1.91	1.73	1.52	1.91
PEF-RTrie	1.28	1.64	1.38	2.15	1.31	2.30
BerkeleyLM C.	1.70 (−8.89%) (+32.90%)	2.83 (+[illegible]) (+72.70%) 2X	1.69 (−11.41%) (+22.04%)	3.48 (+[illegible]) (+61.70%) 2X	1.45 (−4.87%) (+10.83%)	4.13 (+[illegible]) (+79.76%) 2X
BerkeleyLM H.3	6.70 (+258.81%) (+[illegible]) 2.5÷	0.97 (−28.46%) (−40.85%)	7.82 (+310.38%) (+[illegible]) 3.1÷	1.13 (−34.35%) (−47.41%)	9.24 (+[illegible]) (+[illegible]) 5.5X	2.18 (+13.95%) (−5.42%)
BerkeleyLM H.50	7.96 (+[illegible]) (+521.45%) 5.2X	0.97 (−28.49%) (−40.88%)	9.37 (+[illegible]) (+576.87%) 5.8X	0.96 (−44.27%) (−55.35%)	—	—
Expgram	2.06 (+10.18%) (+60.73%)	2.80 (+[illegible]) (+[illegible]) 2X	2.24 (+17.36%) (+61.68%)	9.23 (+[illegible]) (+[illegible]) 3.5X	—	—
KenLM T.	2.99 ([illegible]) (+133.56%) 2.3X	1.28 (−5.47%) (−21.84%)	3.44 ([illegible]) (+148.52%) 2.5X	1.94 (+12.32%) (−10.01%)	—	—
Marisa	3.61 (+93.09%) (+[illegible]) 2.8X	2.06 (+52.00%) (+25.67%)	3.81 (+99.60%) (+[illegible]) 2.7X	3.24 (+87.96%) (+50.58%)	—	—
RandLM	1.81 (−3.06%) (+41.41%)	4.39 (+[illegible]) (+168.04%) 2.5X	2.02 (+6.18%) (+46.29%)	5.08 (+[illegible]) (+135.82%) 2.5X	2.60 (+70.73%) (+98.90%)	9.25 (+[illegible]) (+302.19%) 3X

- Elias-Fano Tries substantially **outperform ALL** previous solutions in **both space and time**.
- As fast as the state-of-the-art (KenLM) but more than twice smaller.

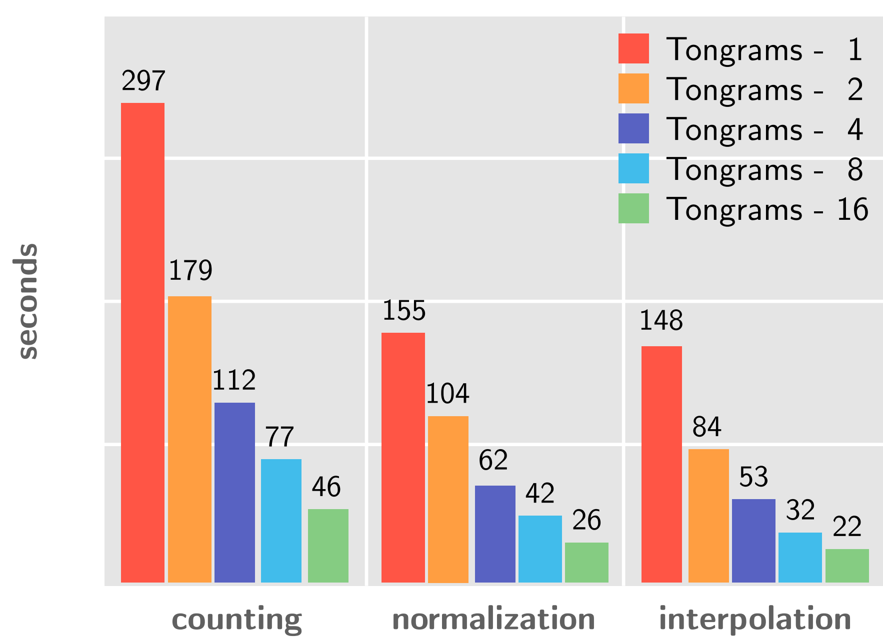
On going work (preliminary results)

Scalable *Modified Kneser-Ney* Language Model Estimation



1.3 GB
233,035,325 total words

1,255,027
20,431,391
82,815,629
153,984,231
196,779,246
— — — — — —
455,265,524

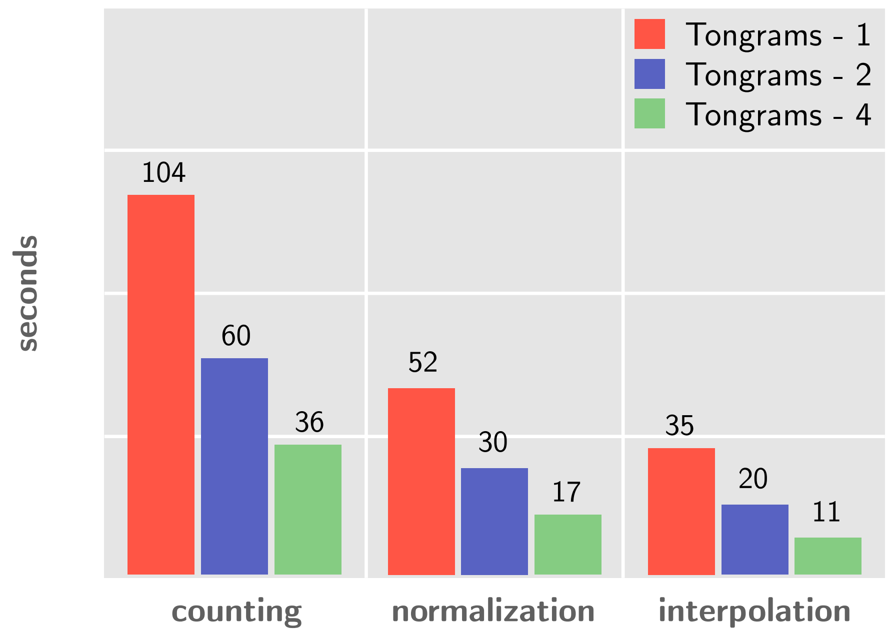


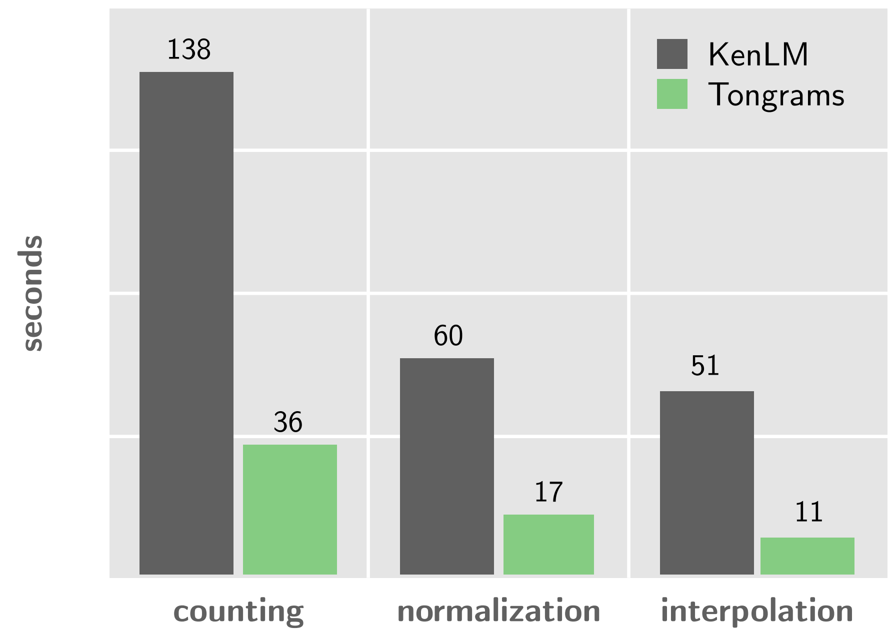
3.2 GB
495,527,349 total words

15,039,323
44,033,774
142,894,817
280,714,113
381,284,741
— — — — — —
863,966,768

On going work (preliminary results)

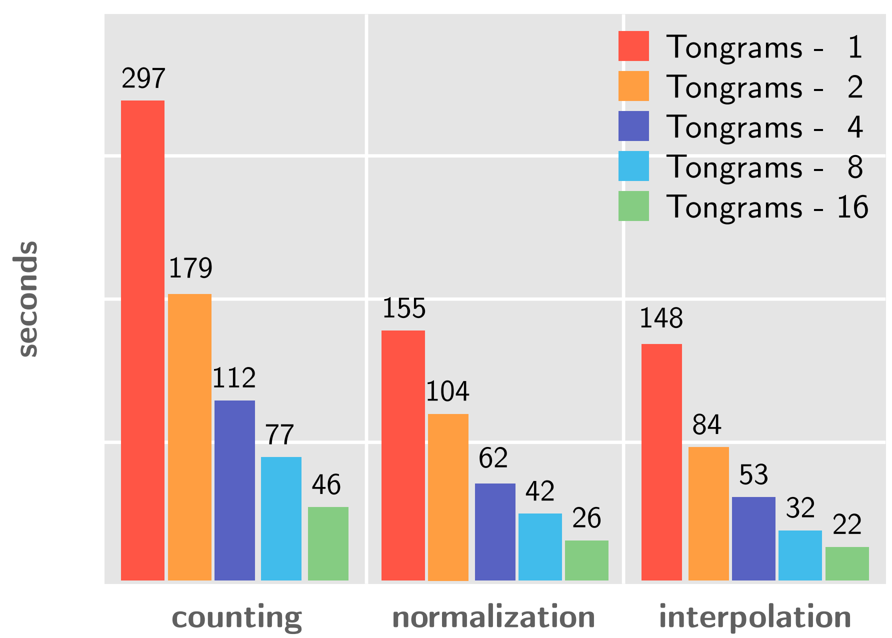
Scalable *Modified Kneser-Ney* Language Model Estimation





1.3 GB
233,035,325 total words

1,255,027
20,431,391
82,815,629
153,984,231
196,779,246
— — — — — —
455,265,524

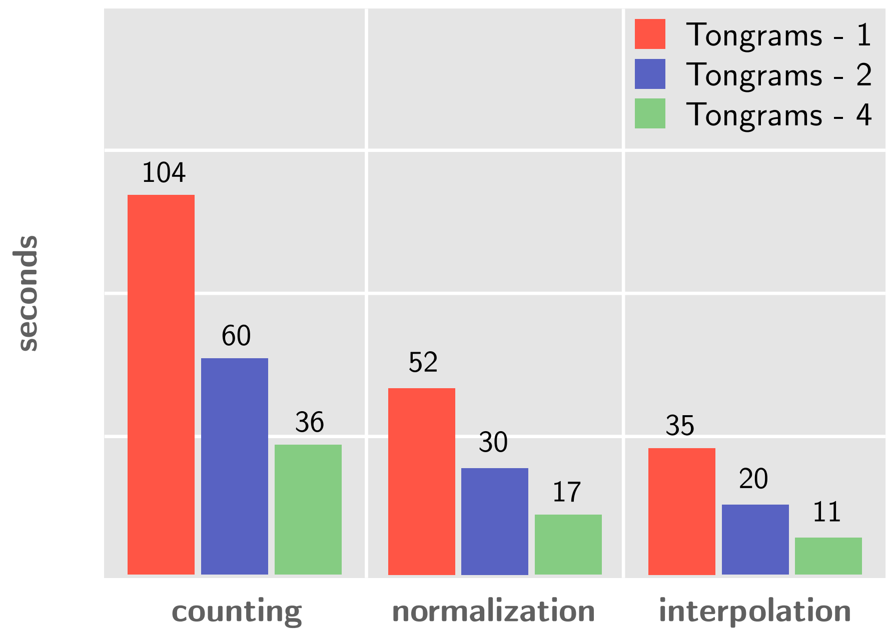


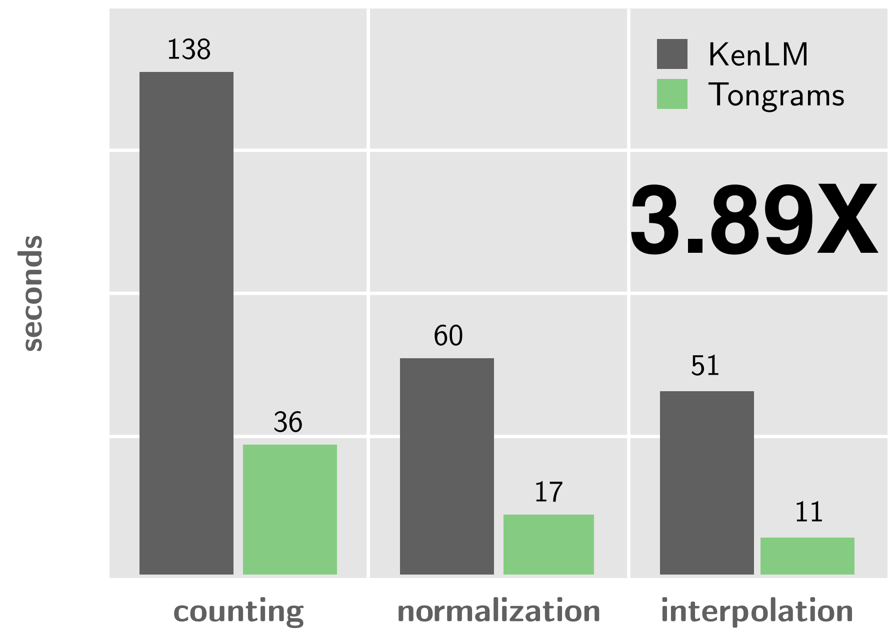
3.2 GB
495,527,349 total words

15,039,323
44,033,774
142,894,817
280,714,113
381,284,741
— — — — — —
863,966,768

On going work (preliminary results)

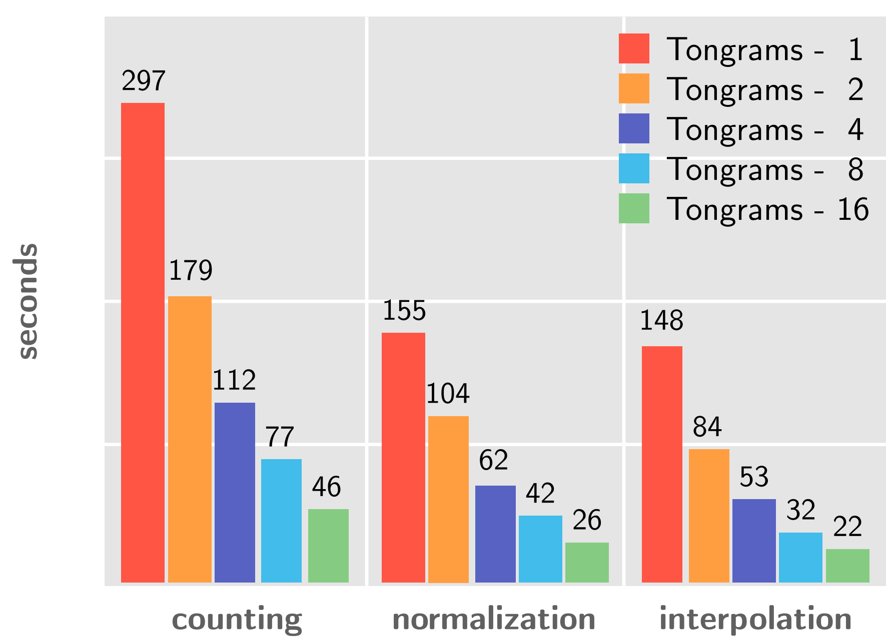
Scalable *Modified Kneser-Ney* Language Model Estimation





1.3 GB
233,035,325 total words

1,255,027
20,431,391
82,815,629
153,984,231
196,779,246
———————
455,265,524

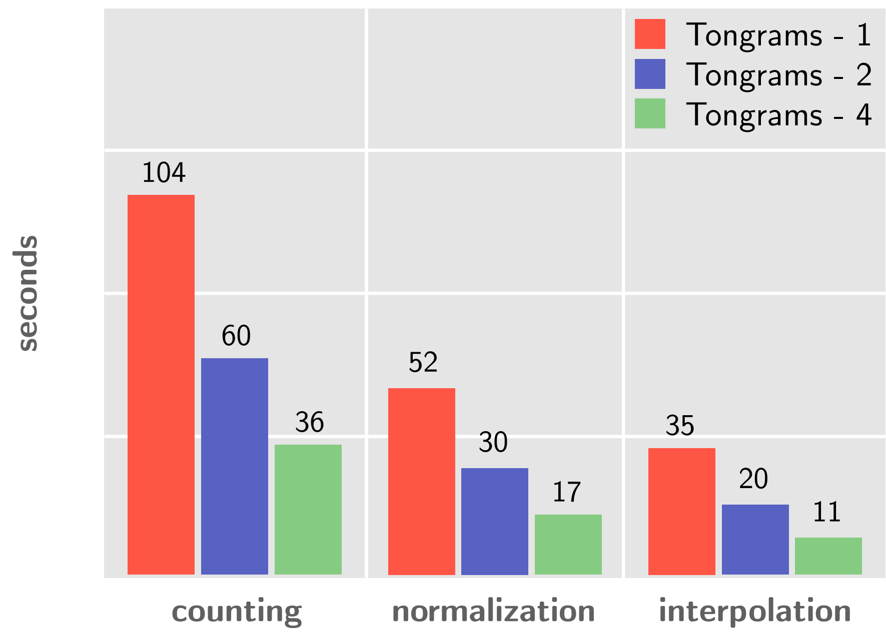


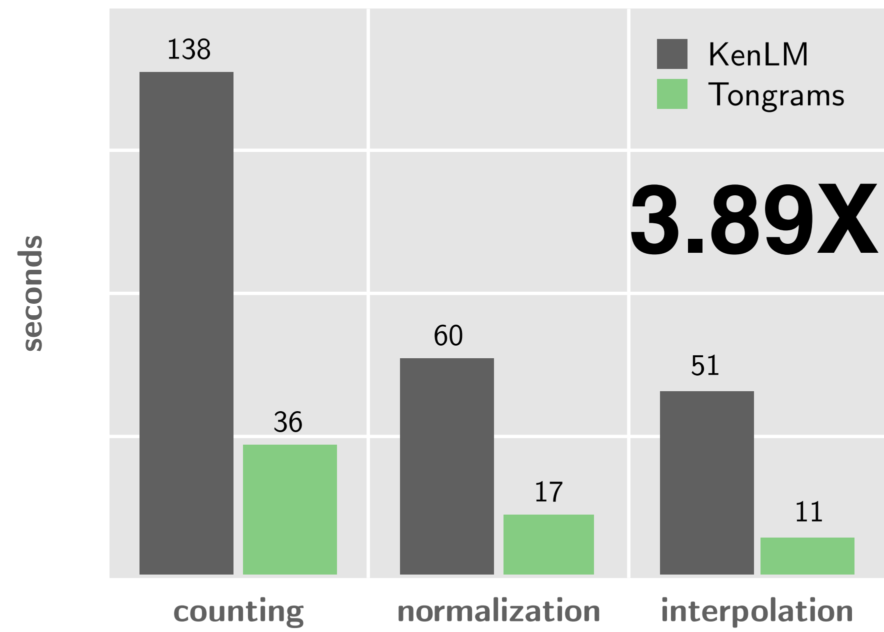
3.2 GB
495,527,349 total words

15,039,323
44,033,774
142,894,817
280,714,113
381,284,741
———————
863,966,768

On going work (preliminary results)

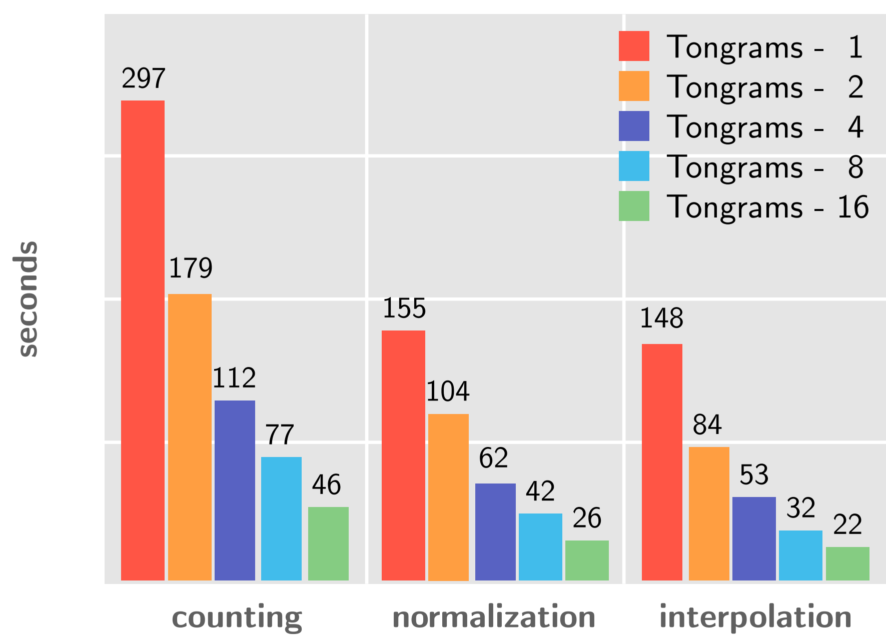
Scalable *Modified Kneser-Ney* Language Model Estimation

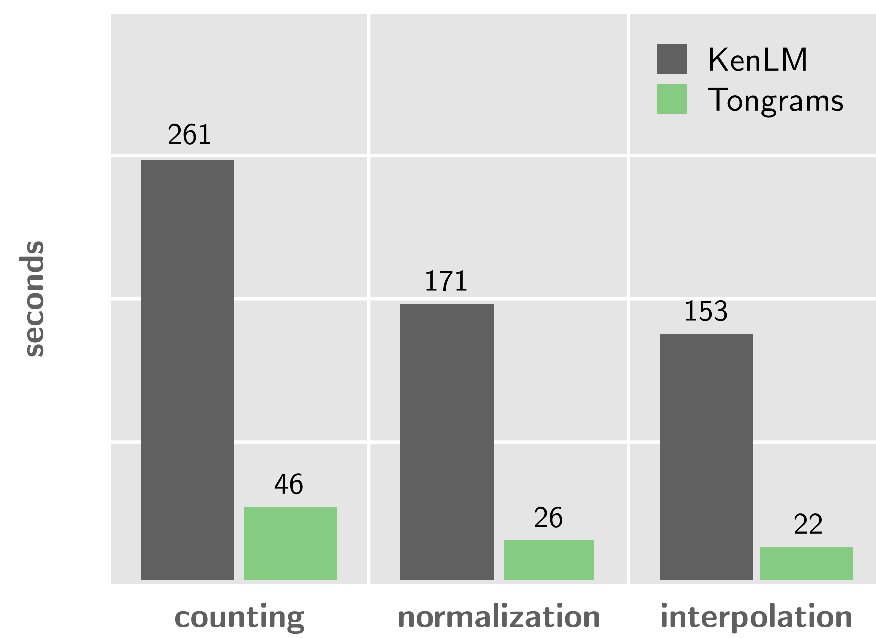




1.3 GB
233,035,325 total words

1,255,027
20,431,391
82,815,629
153,984,231
196,779,246
— — — — — —
455,265,524



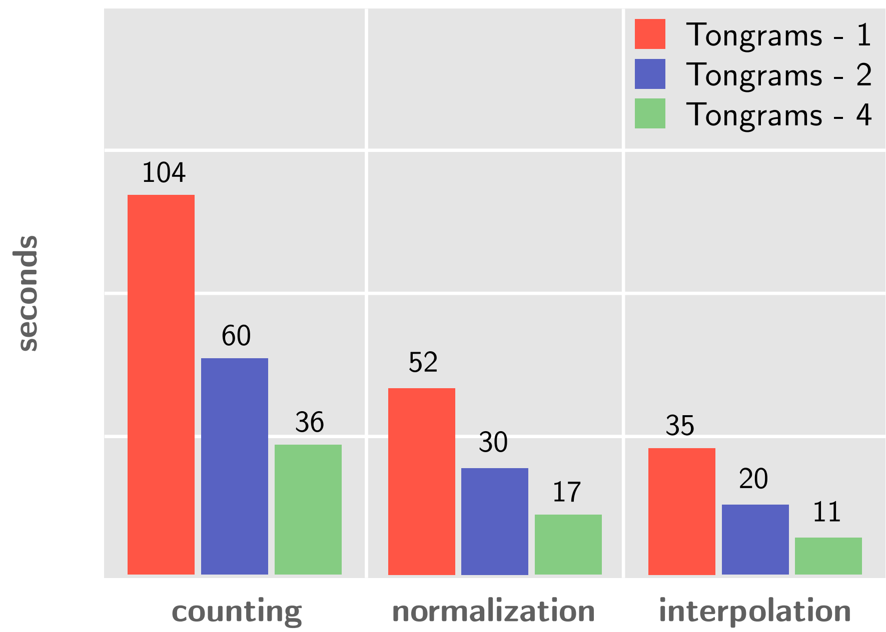


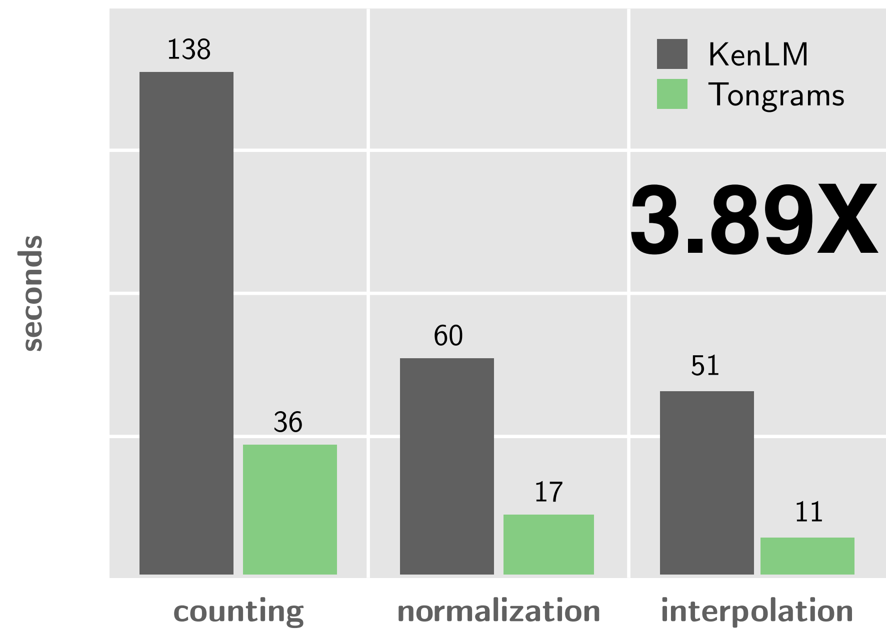
3.2 GB
495,527,349 total words

15,039,323
44,033,774
142,894,817
280,714,113
381,284,741
— — — — — —
863,966,768

On going work (preliminary results)

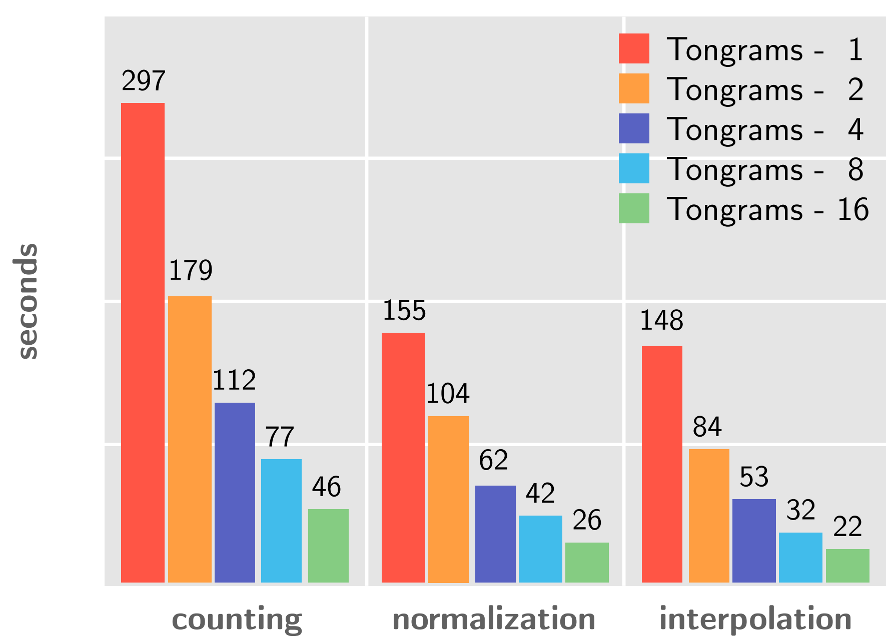
Scalable *Modified Kneser-Ney* Language Model Estimation

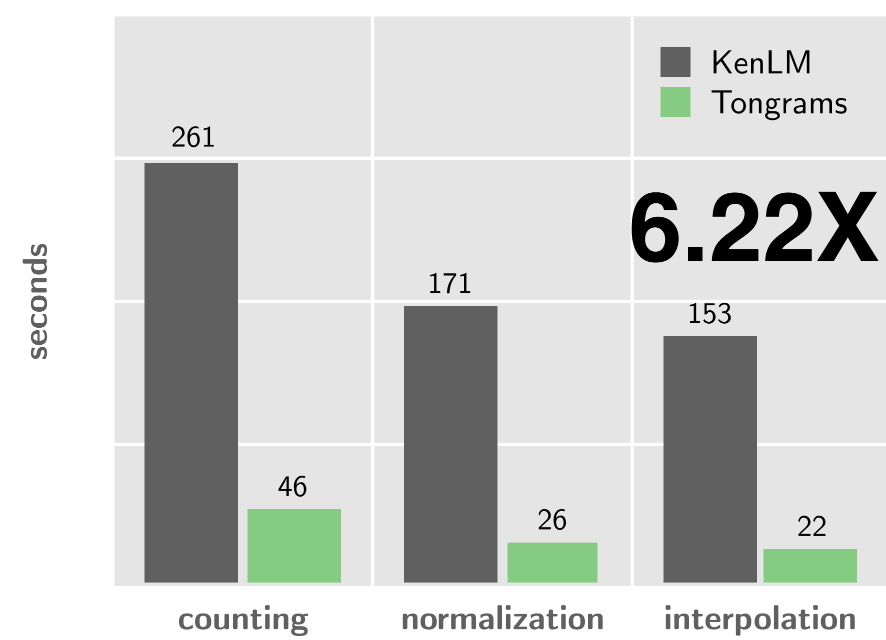




1.3 GB
233,035,325 total words

1,255,027
20,431,391
82,815,629
153,984,231
196,779,246
— — — — — —
455,265,524





3.2 GB
495,527,349 total words

15,039,323
44,033,774
142,894,817
280,714,113
381,284,741
— — — — — —
863,966,768

Planned work for this year

1. Conclude two journal extensions → **TCS**, **VLDBJ**

Planned work for this year

1. Conclude two journal extensions → **TCS** / **VLDBJ**

2. Develop other research ideas

Planned work for this year

1. Conclude two journal extensions → **TCS**, **VLDBJ**

2. Develop other research ideas → Inverted Indexes with *false positives* allowed.

Planned work for this year

1. Conclude two journal extensions
   - **TCS**
   - **VLDBJ**

2. Develop other research ideas
   - Inverted Indexes with *false positives* allowed.
   - Data structures for *features* repository.

Planned work for this year

1. Conclude two journal extensions → **TCS** / **VLDBJ**

2. Develop other research ideas →
   - Inverted Indexes with *false positives* allowed.
   - Data structures for *features* repository.
   - Compressed tries based on *double-arrays*.

Planned work for this year

1. Conclude two journal extensions → **TCS**, **VLDBJ**

2. Develop other research ideas →
   - Inverted Indexes with *false positives* allowed.
   - Data structures for *features* repository.
   - Compressed tries based on *double-arrays*.

3. 6 months abroad.

Thanks for your attention, time, patience!

Any questions?