Efficient Data Structures for Massive *N*-Gram Datasets

Giulio Ermanno Pibiri
University of Pisa and ISTI-CNR
Pisa, Italy
giulio.pibiri@di.unipi.it

Rossano Venturini
University of Pisa and ISTI-CNR
Pisa, Italy
rossano.venturini@unipi.it

The 40-th ACM SIGIR Conference
on Research and Development in Information Retrieval

Tokyo, Japan

10/08/2017

N-grams - Introduction

Strings of *N* words.

N typically ranges from 1 to 5.

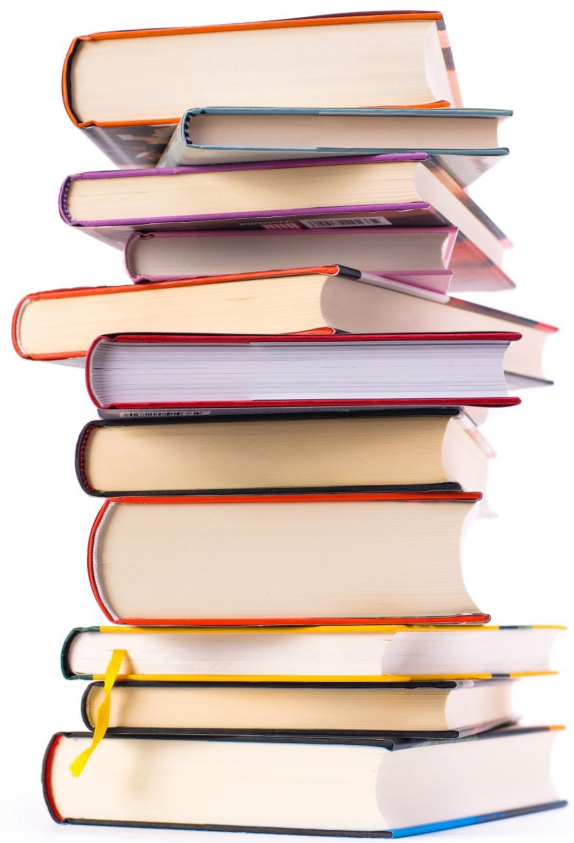
Extracted from text using a *sliding window* approach.

N-grams - Introduction

Strings of *N* words.

N typically ranges from 1 to 5.

Extracted from text using a *sliding window* approach.

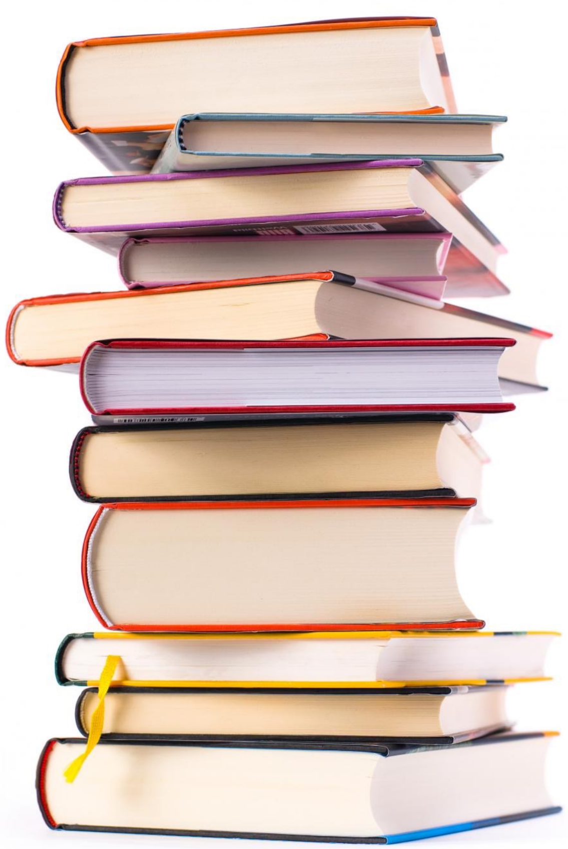


N-grams - Introduction

Strings of *N* words.

N typically ranges from 1 to 5.

Extracted from text using a *sliding window* approach.



Google Books

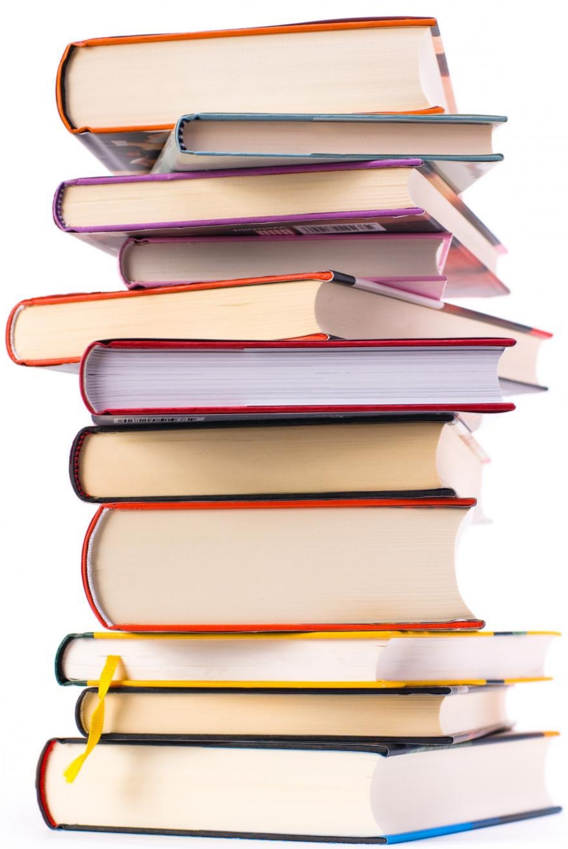
≈ 6% of the books ever published

N-grams - Introduction

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.

N-grams - Challenge

Store massive *N*-grams datasets in **compressed space** such that given a pattern, we can **return its value efficiently**.

N-grams - Challenge

Store massive *N*-grams datasets in **compressed space** such that given a pattern, we can **return its value efficiently**.

N-Gram **values**

N-grams - Challenge

Store massive *N*-grams datasets in **compressed space** such that given a pattern, we can **return its value efficiently**.

N-Gram **values**

frequency count
(integer)

N-grams - Challenge

Store massive *N*-grams datasets in **compressed space** such that given a pattern, we can **return its value efficiently**.

N-Gram **values** → frequency count (integer)

N-Gram **values** → probability/backoff weight (floating point)

For backoff-interpolated models, such as *Kneser-Ney*.

N-grams - Challenge

Store massive *N*-grams datasets in **compressed space** such that given a pattern, we can **return its value efficiently**.

N-Gram **values**

frequency count
(integer)

probability/backoff
weight
(floating point)

For backoff-interpolated models, such as *Kneser-Ney*.

Efficient **map**

N-grams - Challenge

Store massive *N*-grams datasets in **compressed space** such that given a pattern, we can **return its value efficiently**.

N-Gram **values**

frequency count
(integer)

probability/backoff weight
(floating point)

For backoff-interpolated models, such as *Kneser-Ney*.

Efficient **map**

hash

+ time
- space

N-grams - Challenge

Store massive *N*-grams datasets in **compressed space** such that given a pattern, we can **return its value efficiently**.

N-Gram **values**

- frequency count (integer)
- probability/backoff weight (floating point)

For backoff-interpolated models, such as *Kneser-Ney*.

Efficient **map**

- **hash**
  + \+ time
  + \- space
- **trie**
  + \+ space
  + \- time

N-grams - Challenge

Store massive *N*-grams datasets in **compressed space** such that given a pattern, we can **return its value efficiently**.

N-Gram **values**

- frequency count (integer)
- probability/backoff weight (floating point)

For backoff-interpolated models, such as *Kneser-Ney*.

Efficient **map**

- **hash**: + time, - space
- **trie**: + space, - time

Active field of research
Many software libraries

- **KenLM** [Heafield, WMT 2011]
- **BerkeleyLM** [Pauls and Klein, ACL 2011]
- **ExpGram** [Watanabe at el., IJCNLP 2009]
- **IRSTLM** [Federico et al., ACL 2008]
- **RandLM** [Talbot and Osborne, ACL 2007]
- **SRILM** [Stolcke, INTERSPEECH 2002]

Trie Indexing

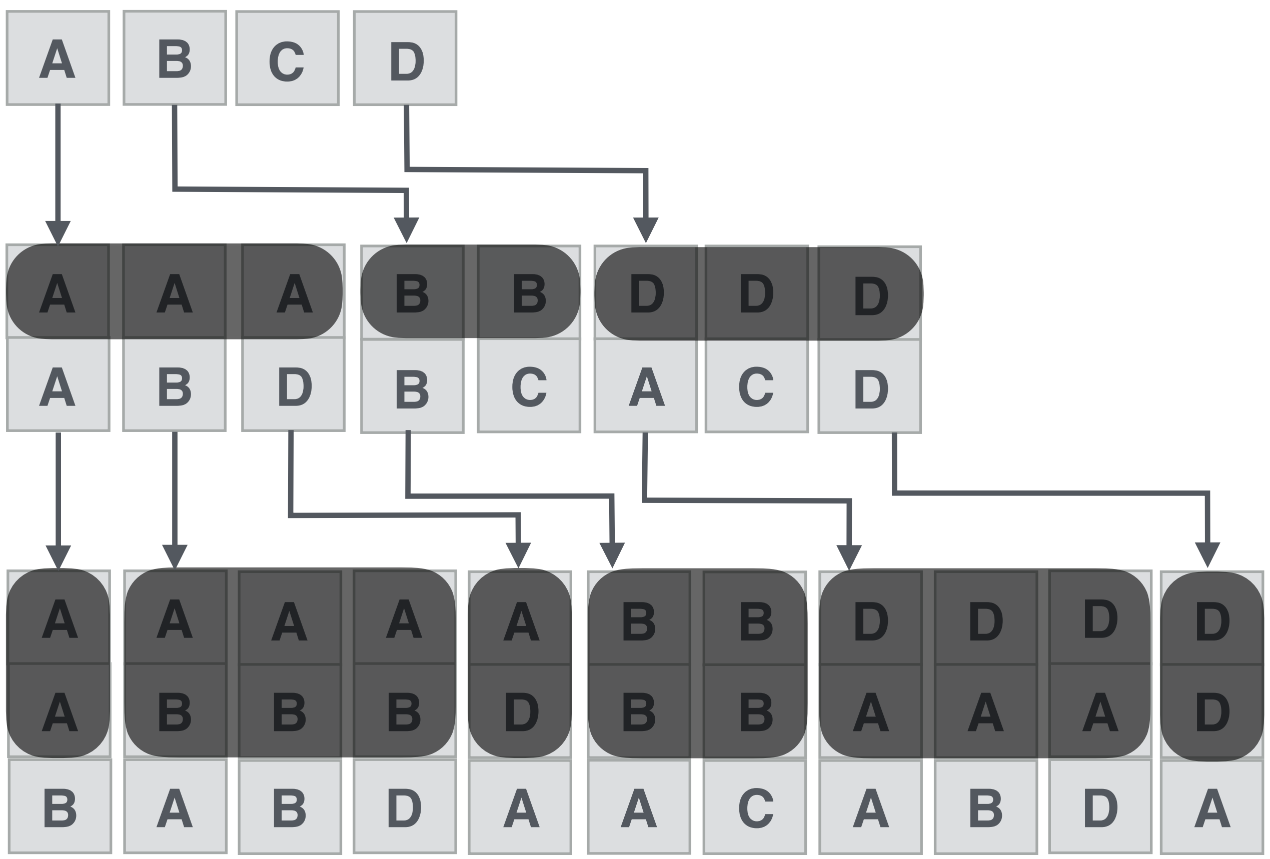
Trie Indexing

A	B	C	D

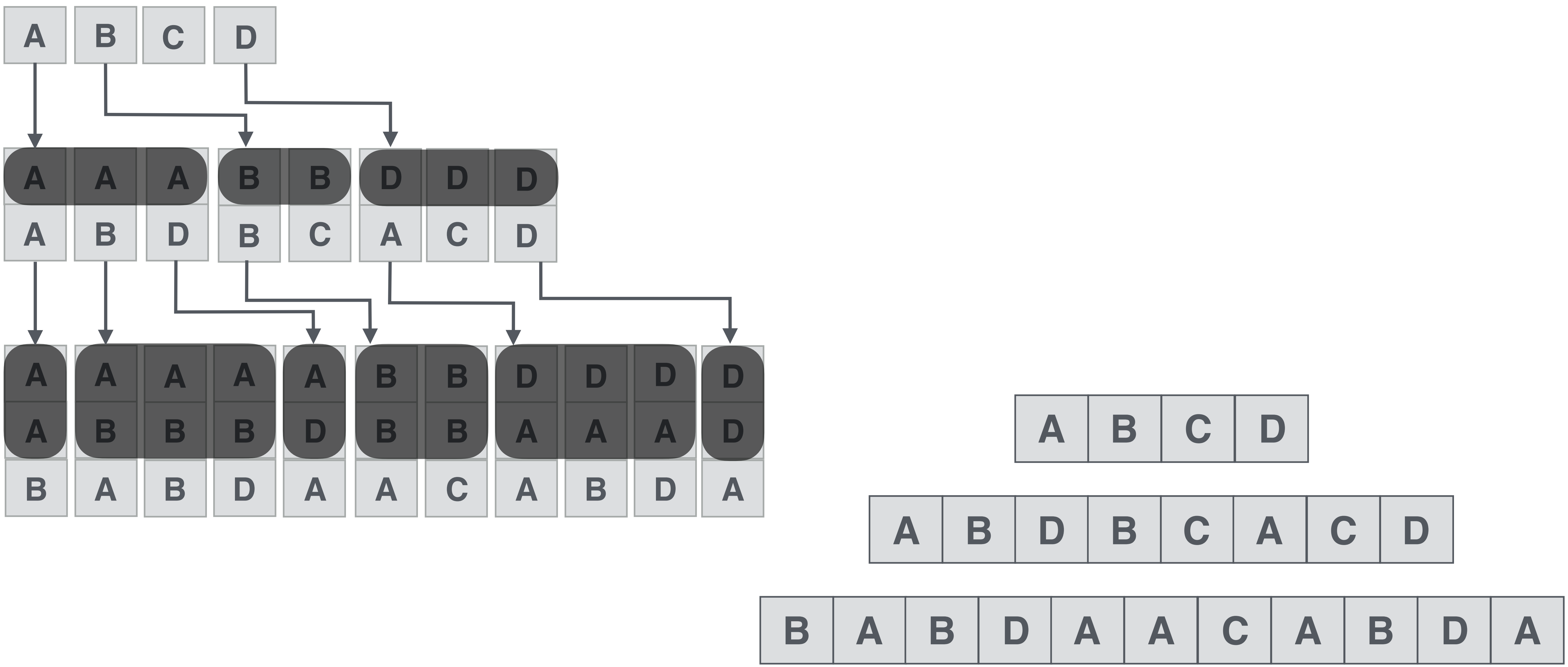
A	A	A	B	B	D	D	D
A	B	D	B	C	A	C	D

A	A	A	A	A	B	B	D	D	D	D
A	B	B	B	D	B	B	A	A	A	D
B	A	B	D	A	A	C	A	B	D	A

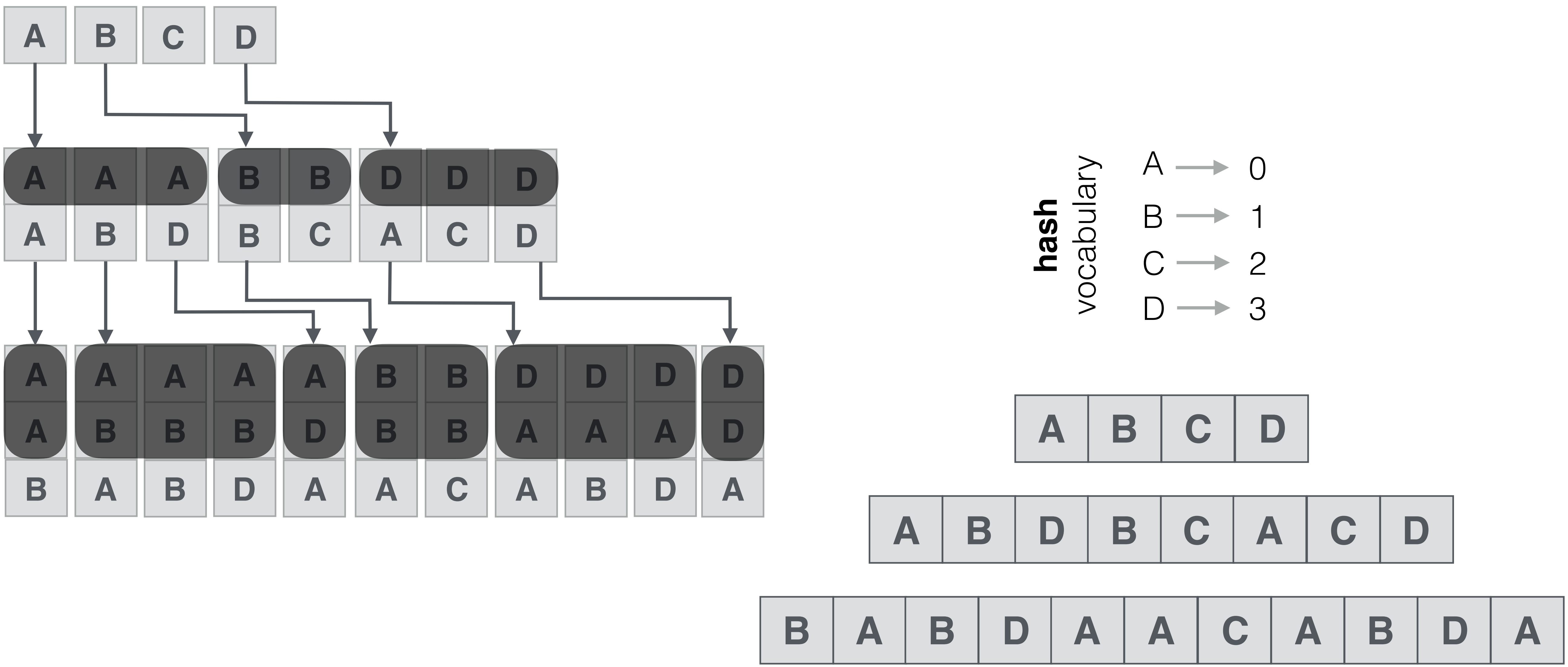
Trie Indexing



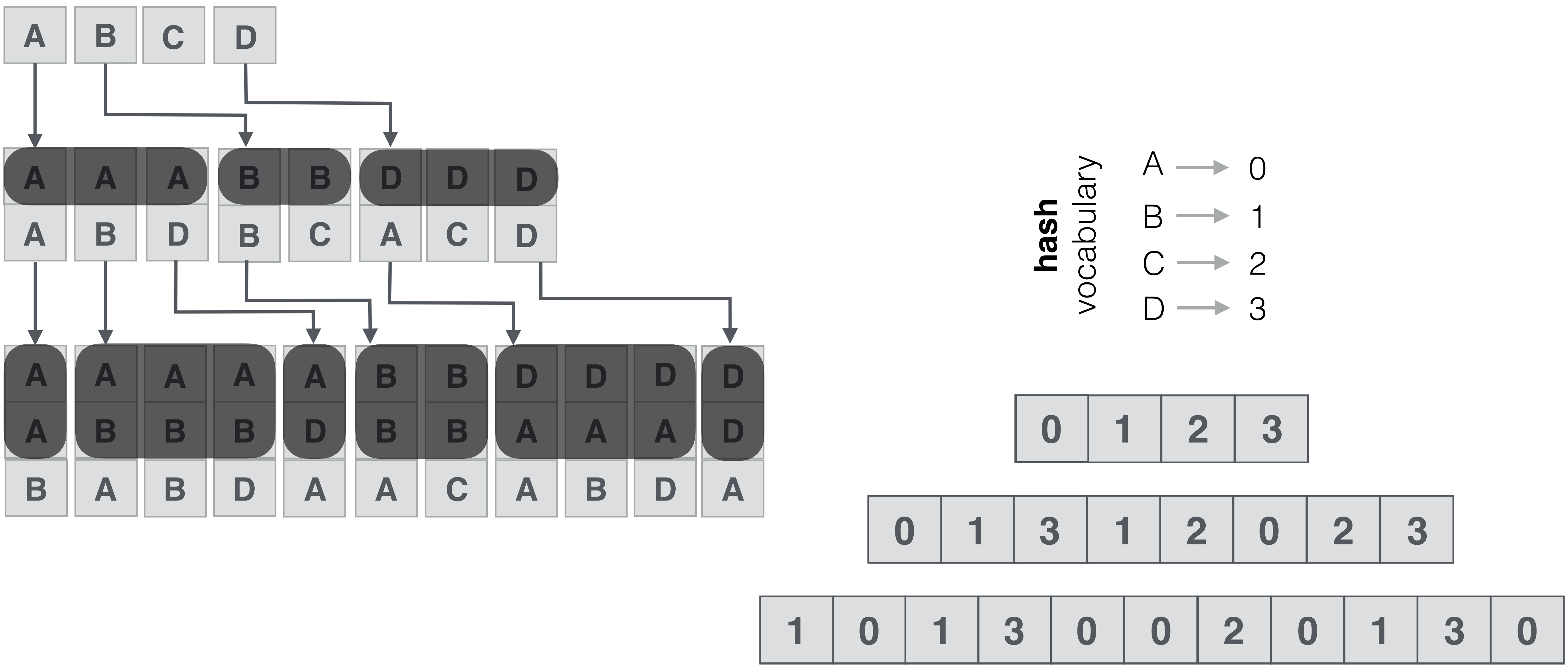
Trie Indexing



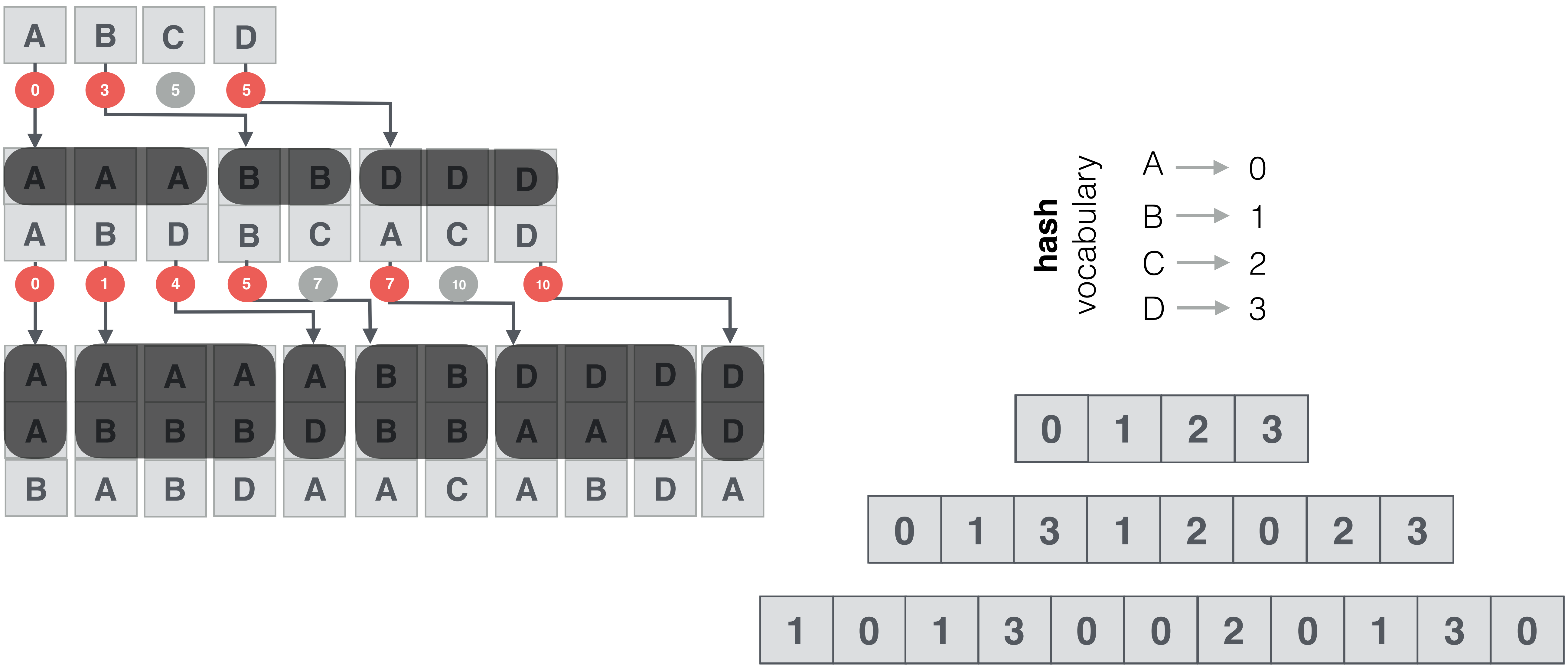
Trie Indexing



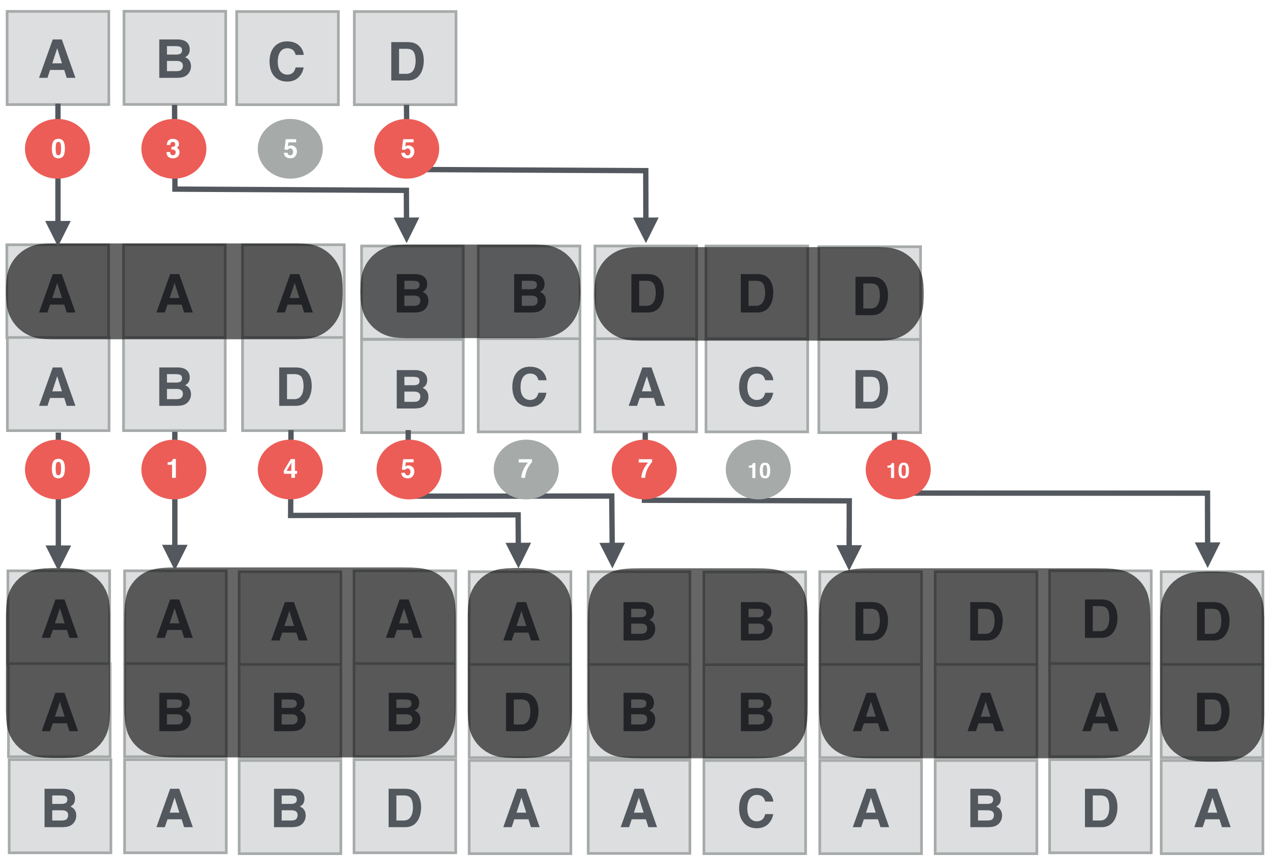
Trie Indexing

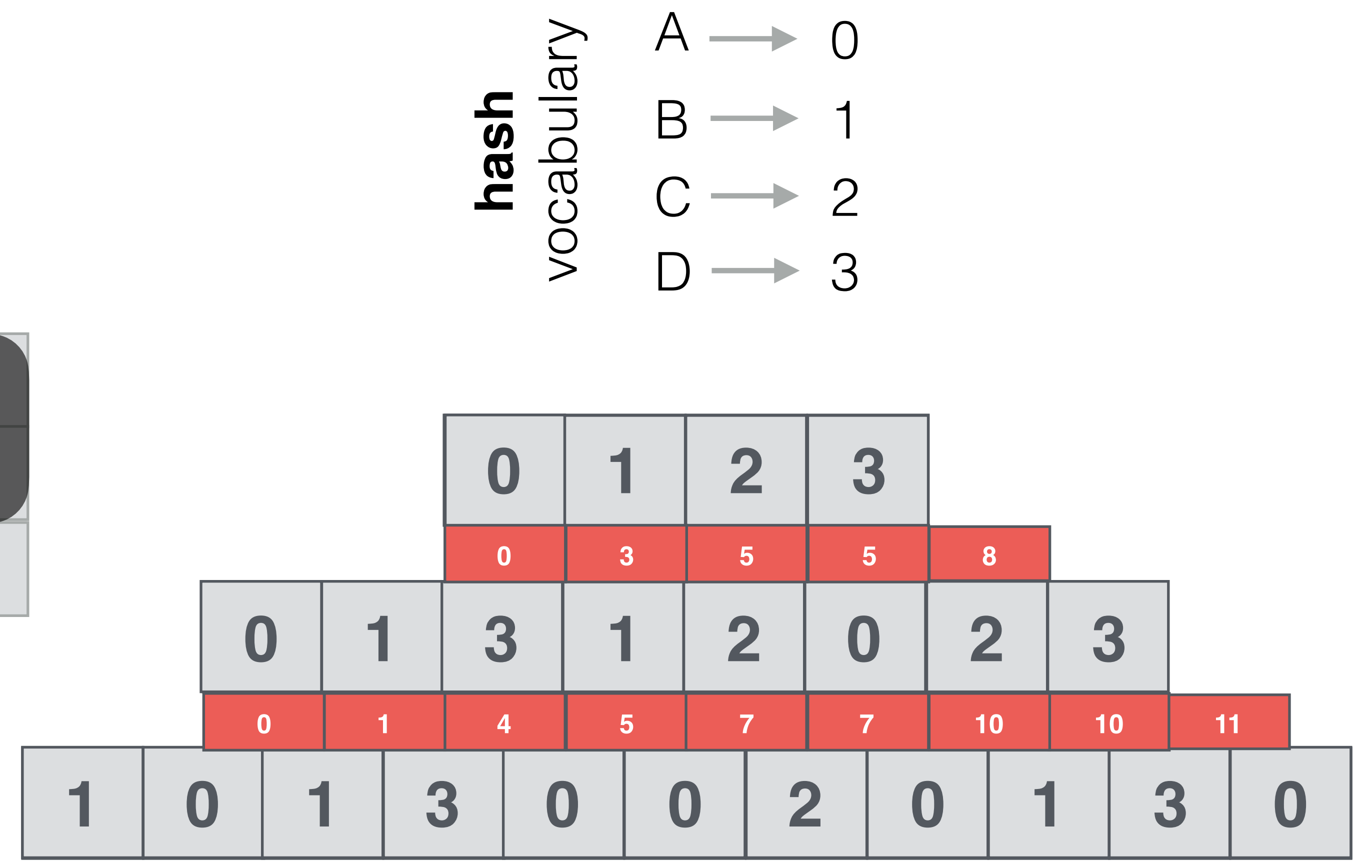


Trie Indexing

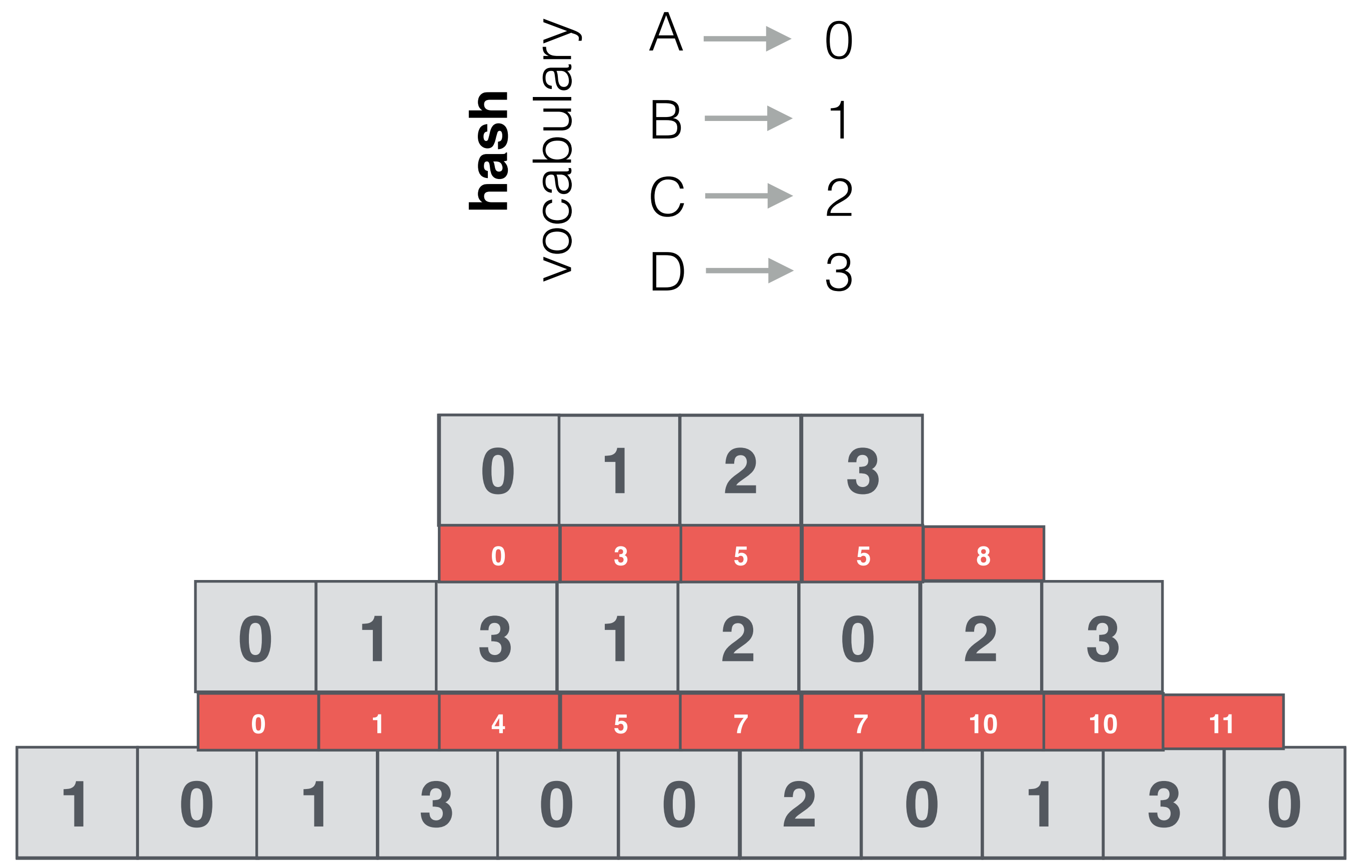


Trie Indexing



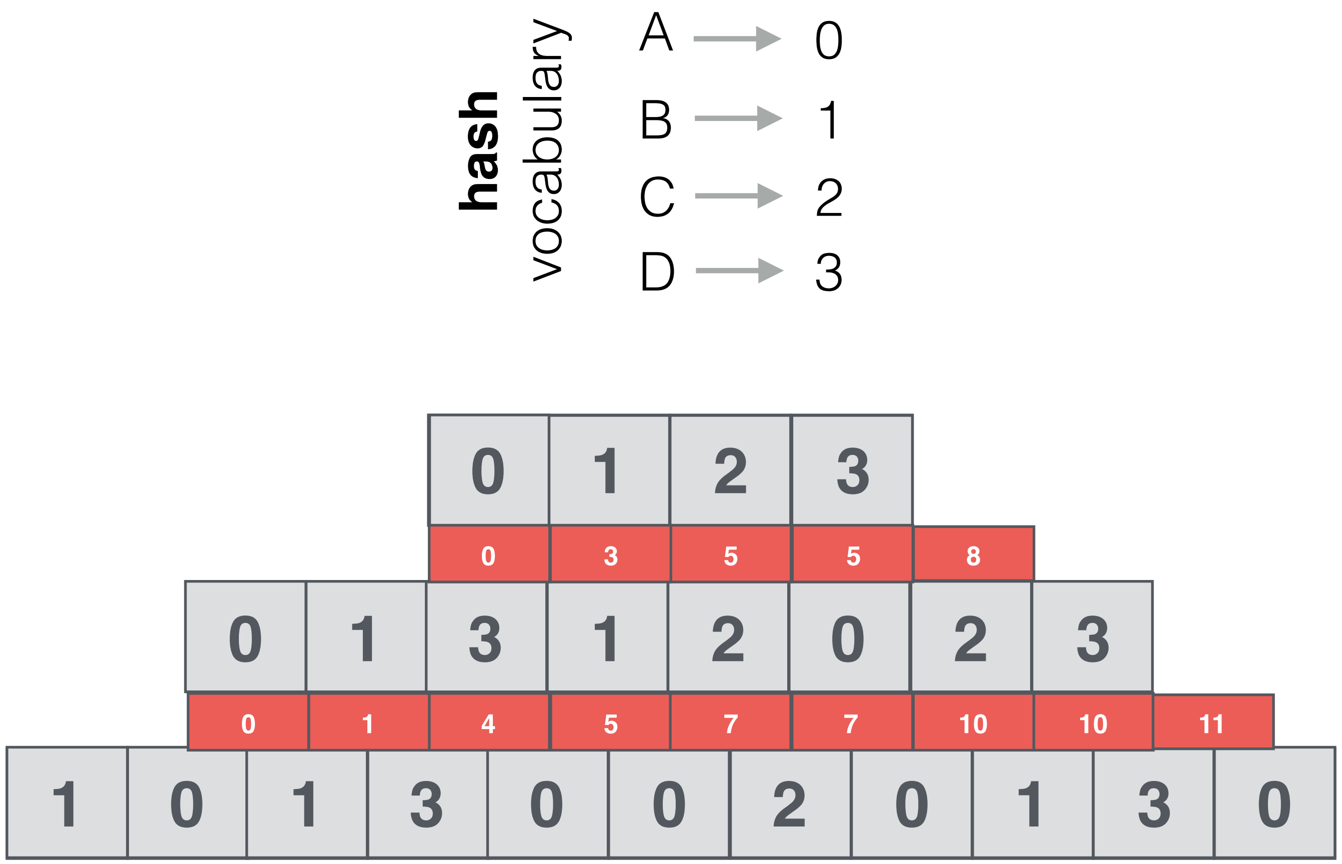


Trie Indexing



Trie Indexing

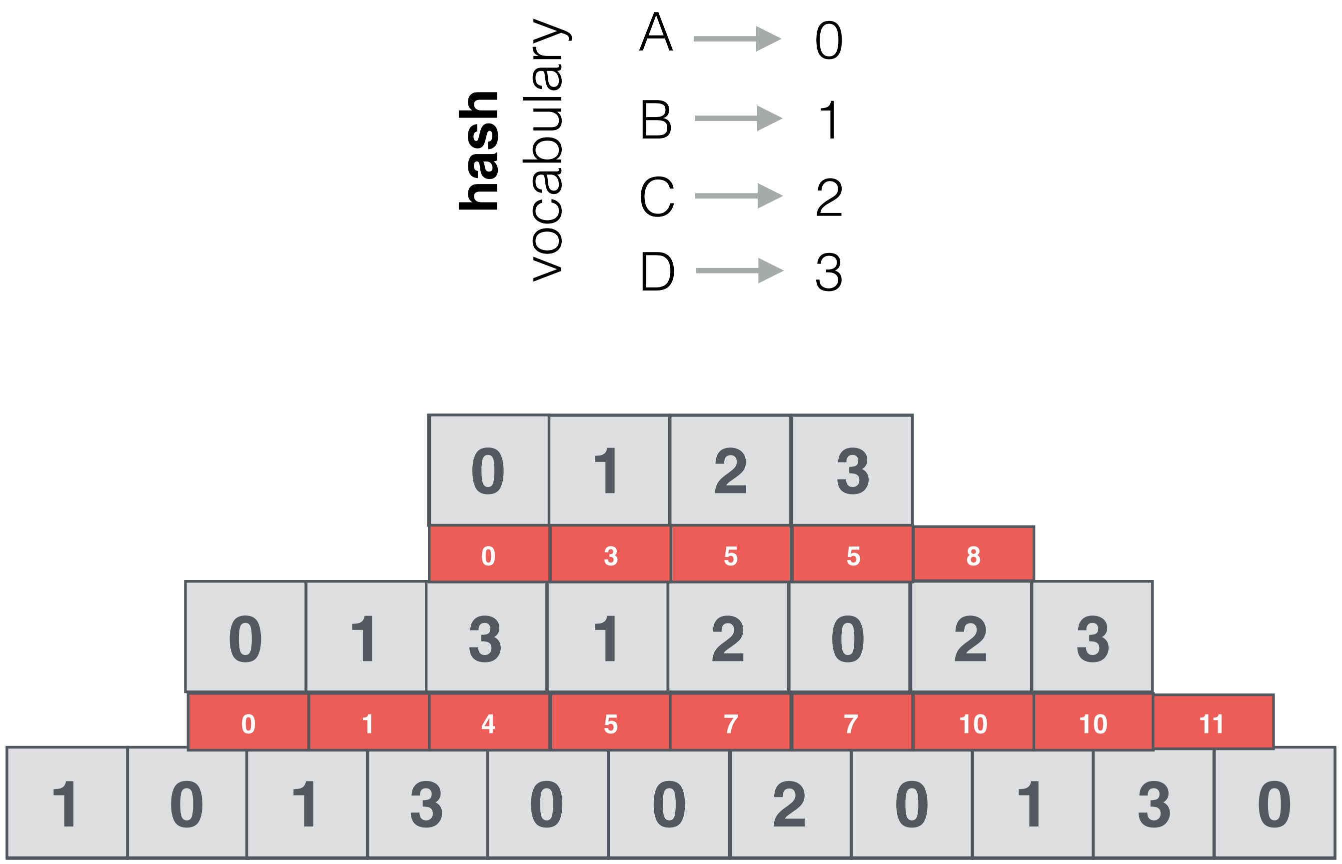
We need an encoder for integer sequences, supporting fast **random** Access.



Trie Indexing

We need an encoder for integer sequences, supporting fast **random** Access.

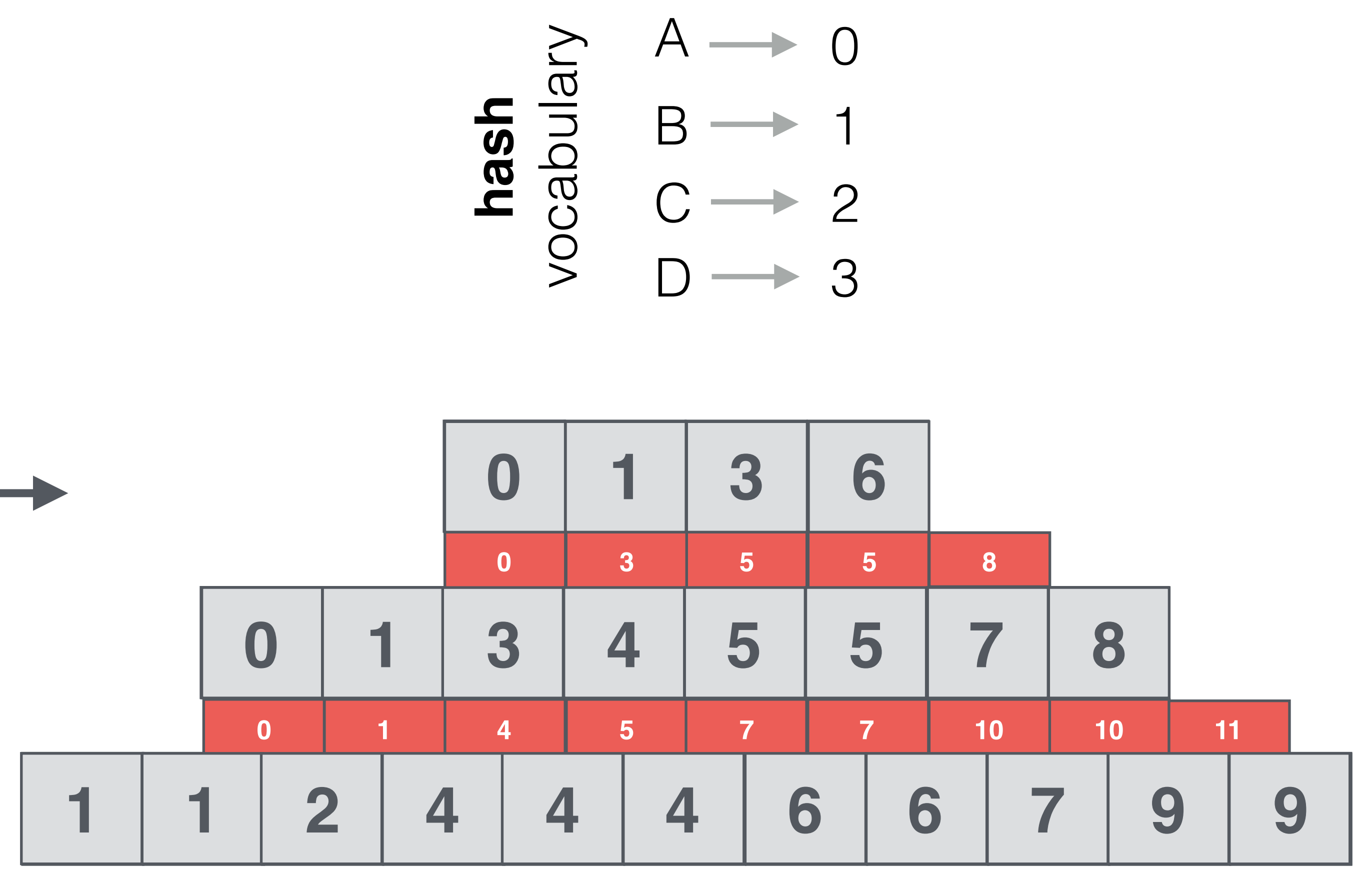
Take *range-wise* prefix sums on gram-ID sequences.



Trie Indexing

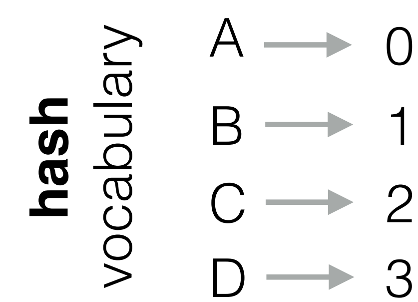
We need an encoder for integer sequences, supporting fast **random Access.**

Take *range-wise* prefix sums on gram-ID sequences.

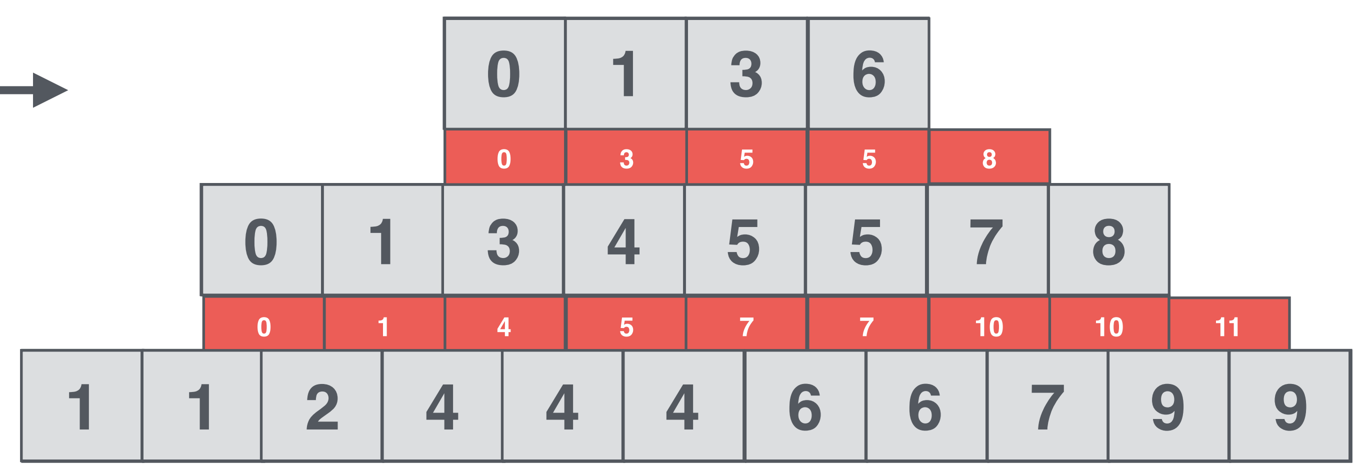


Trie Indexing

We need an encoder for integer sequences, supporting fast **random** `Access`**.**



Take *range-wise* prefix sums on gram-ID sequences.



Elias-Fano Tries

One `NextGEQ` per level

Constant-time random `Access`

Context-based ID Remapping

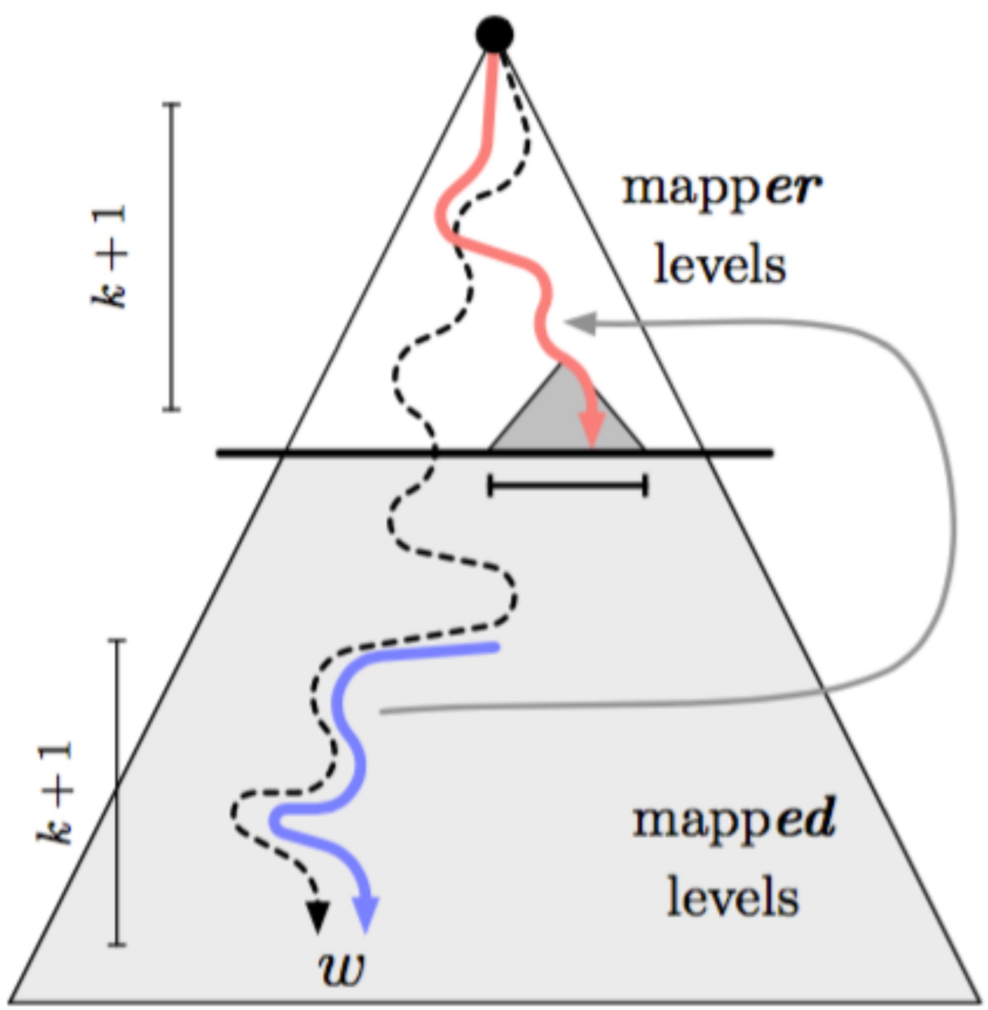
Observation: the number of words following a given context is **small**.

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).

Context-based ID Remapping

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

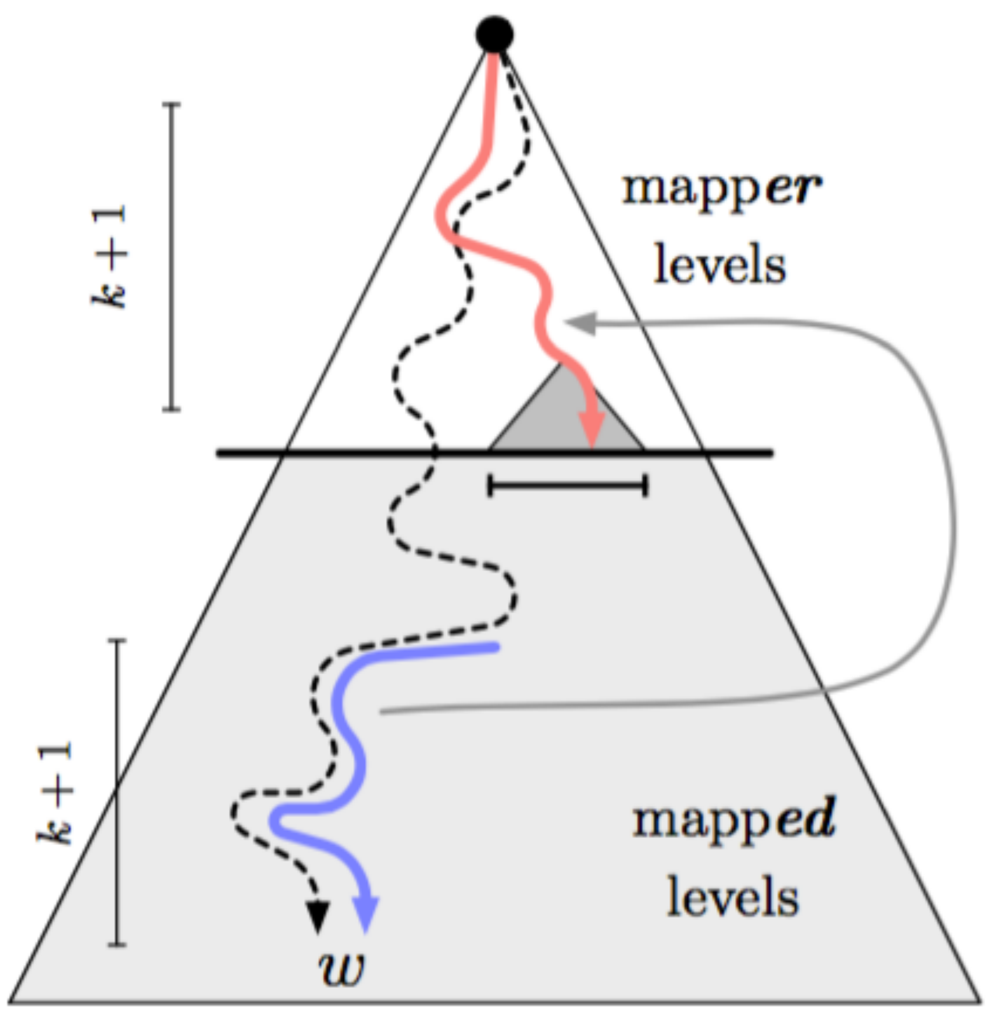
High-level idea: map a word ID to the **position** it takes within its *sibling* IDs (the IDs following a context of fixed length *k*).

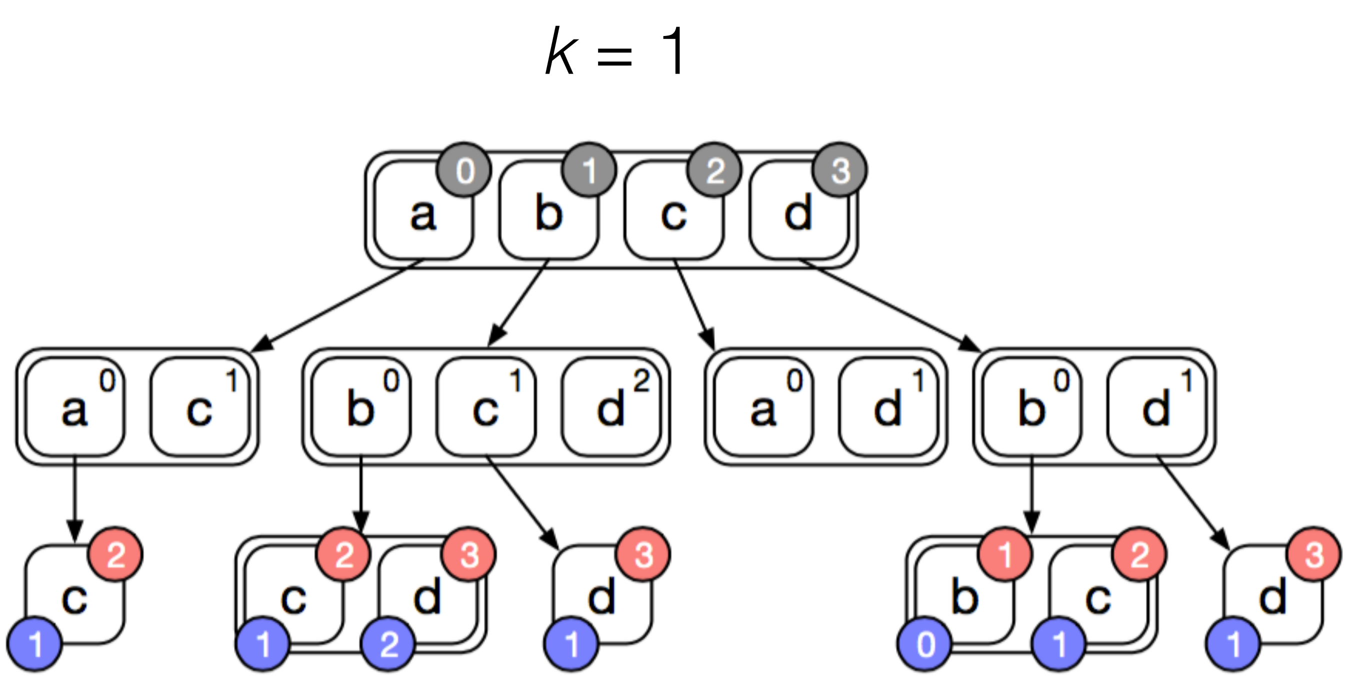


Context-based ID Remapping

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).

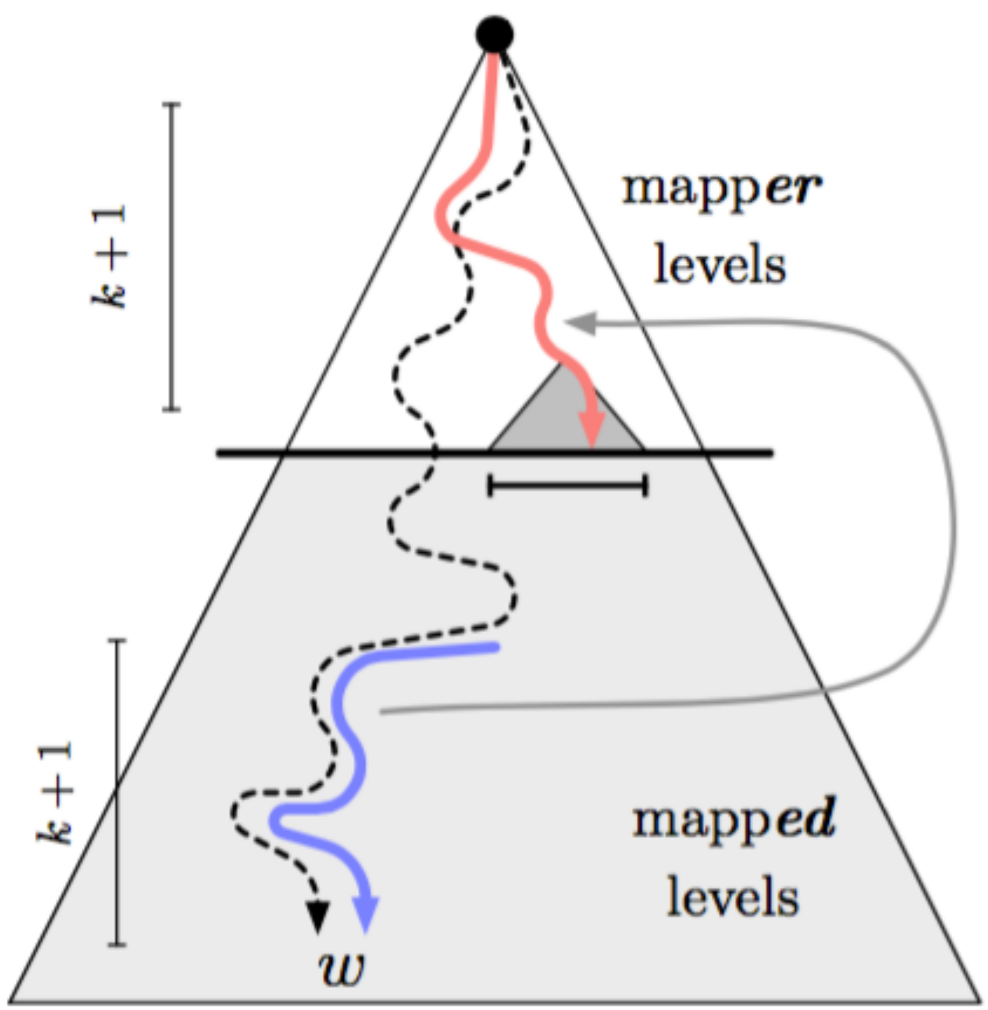


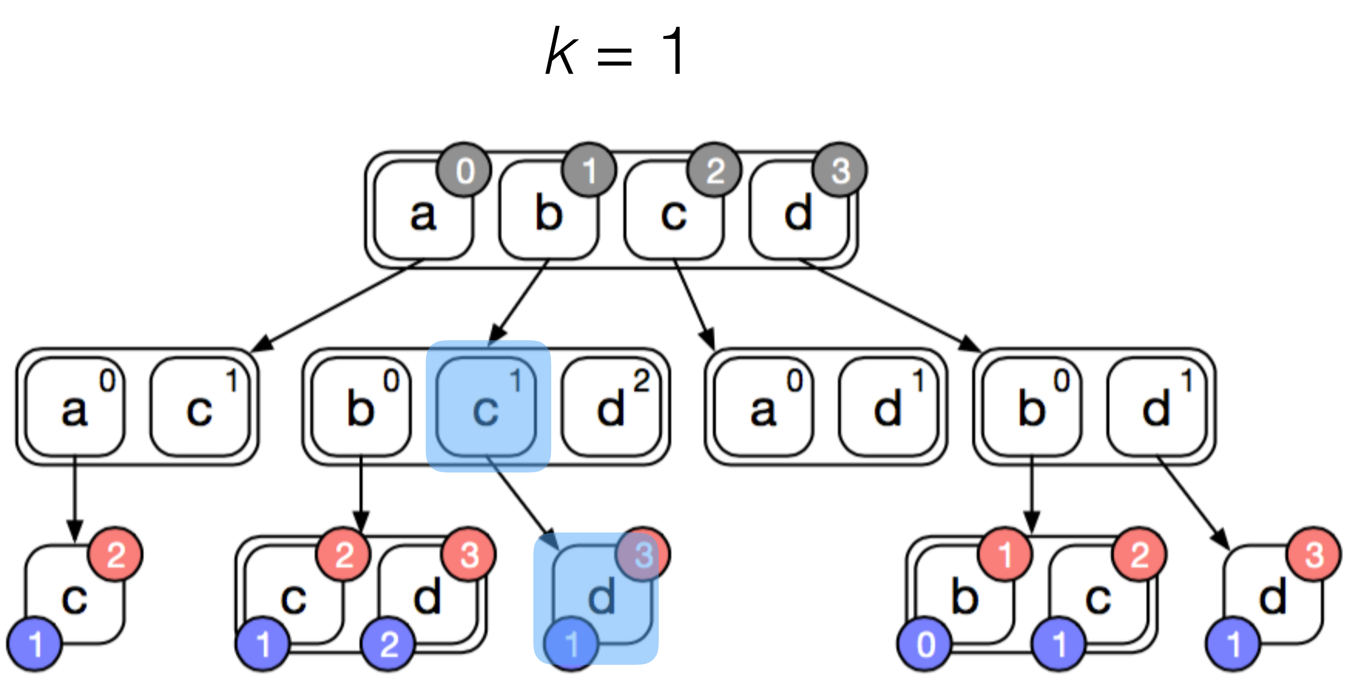


Context-based ID Remapping

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*).

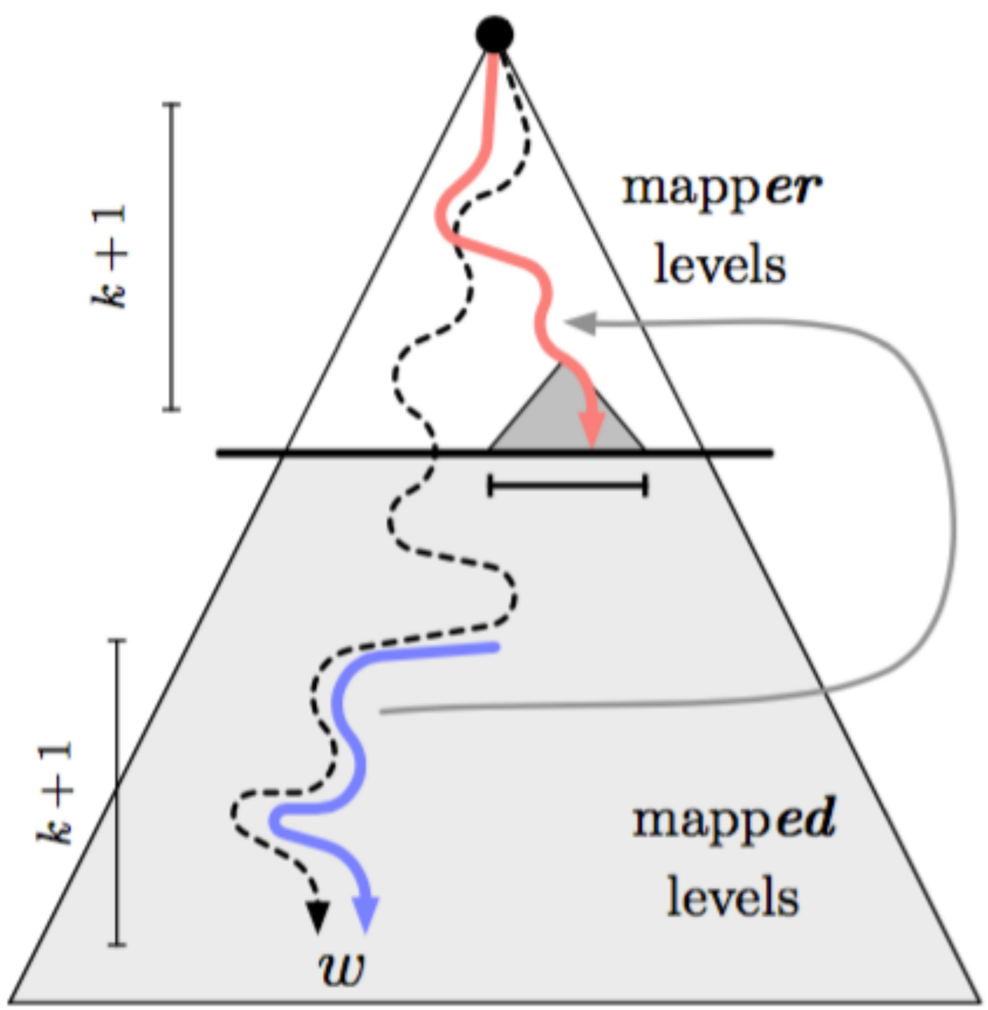


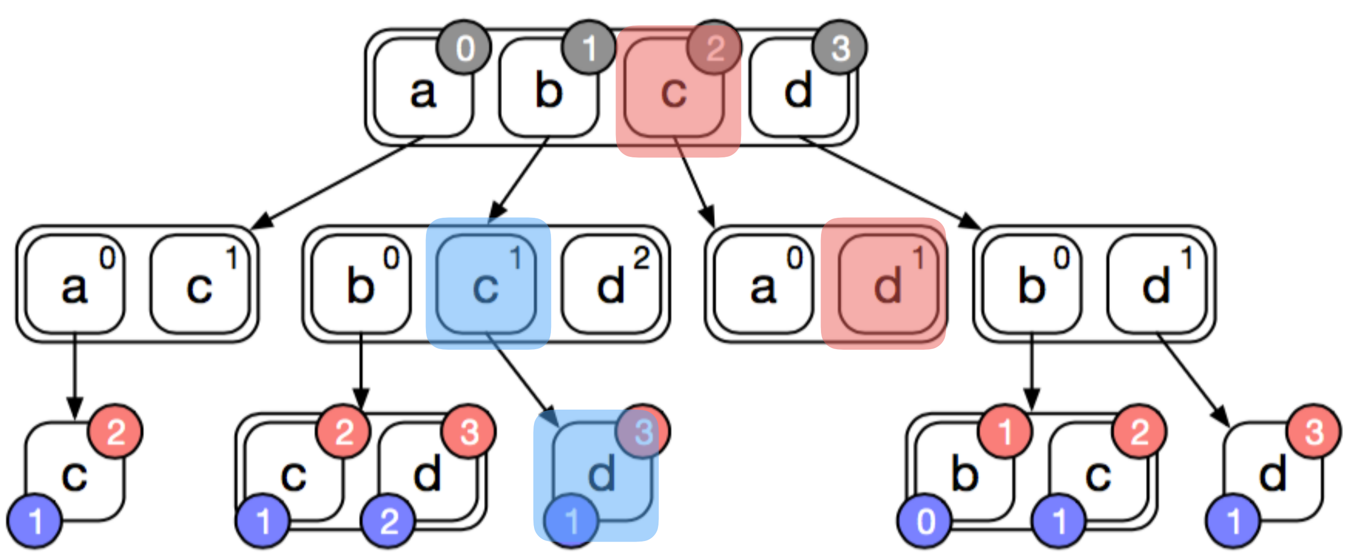


Context-based ID Remapping

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*).

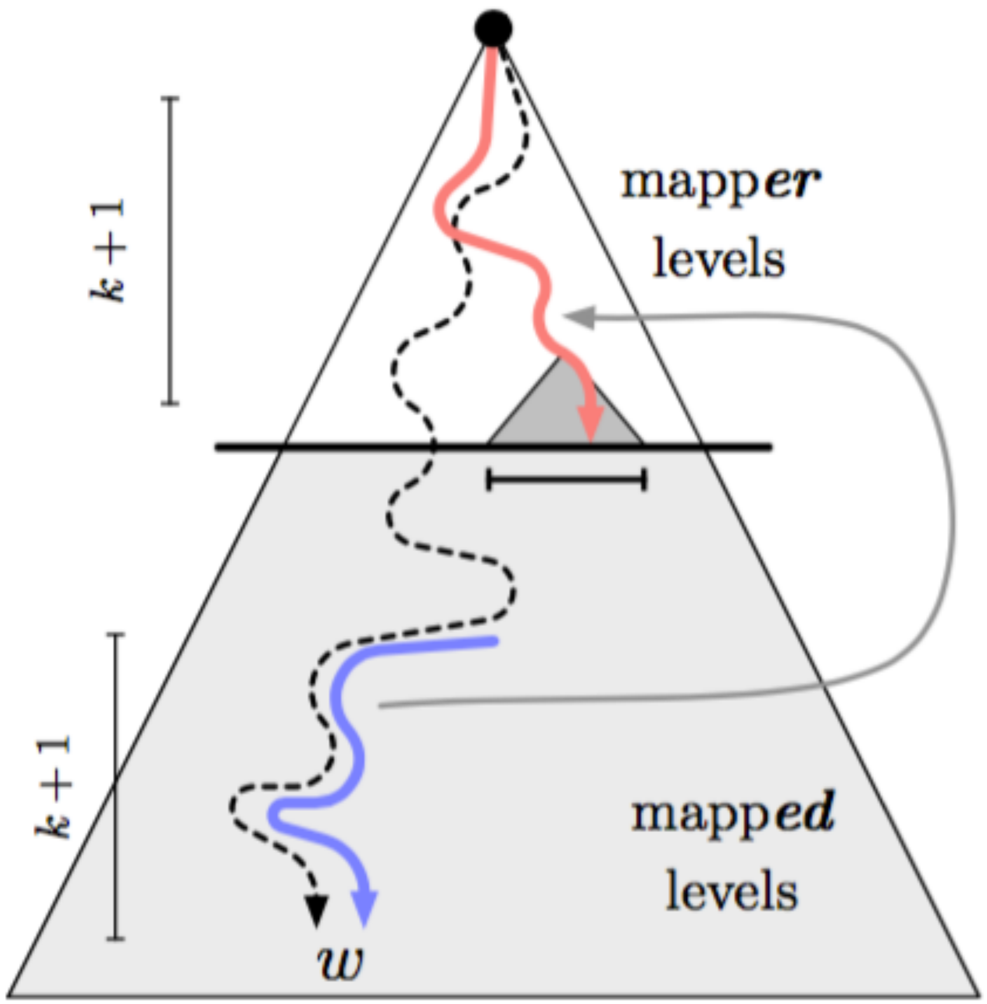


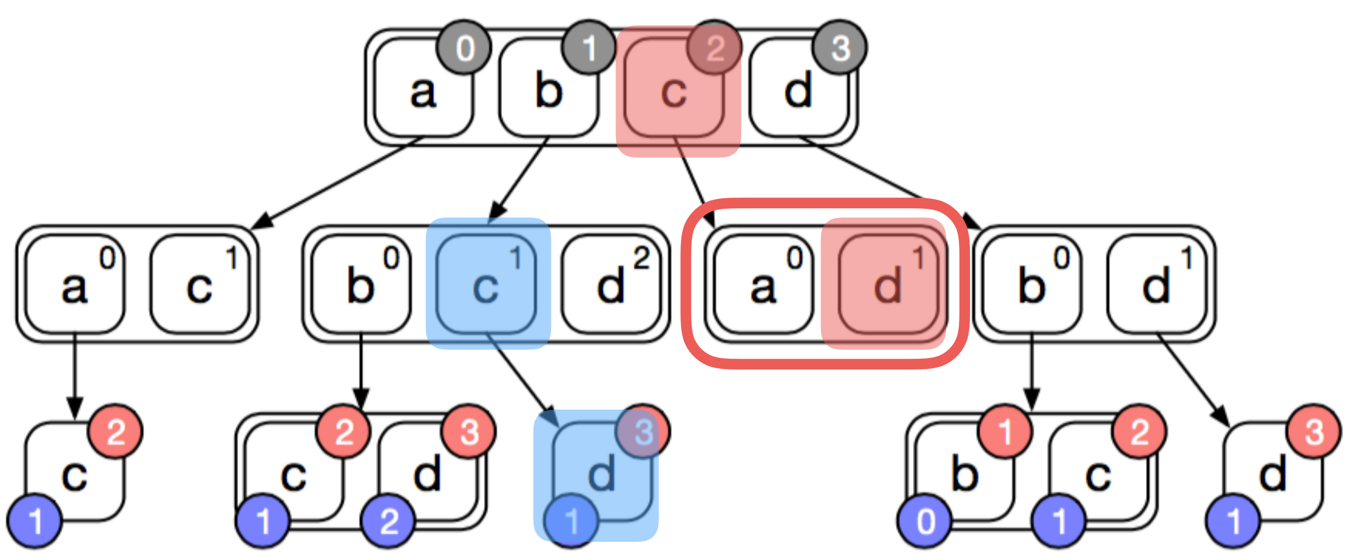


Context-based ID Remapping

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*).





Context-based ID Remapping

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).

Context-based ID Remapping

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*).

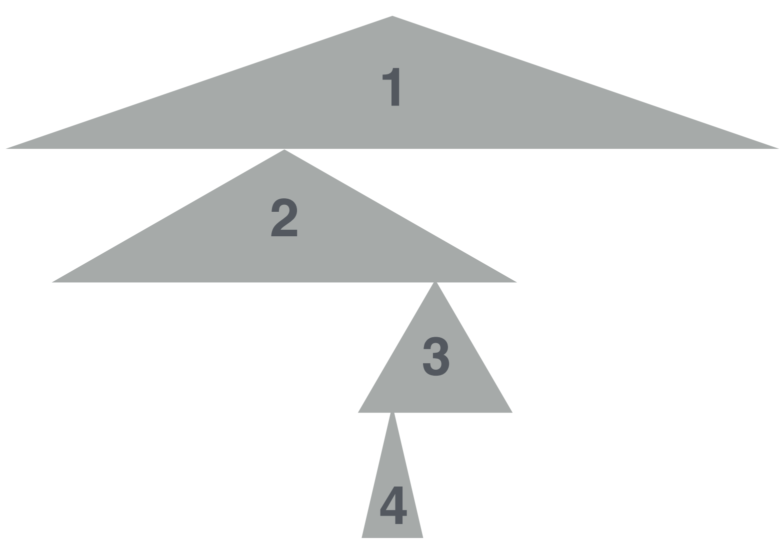
- **Millions** of unigrams.
- Height 5: **longer** contexts.
- The number of siblings has a **funnel-**shaped distribution.

Context-based ID Remapping

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.

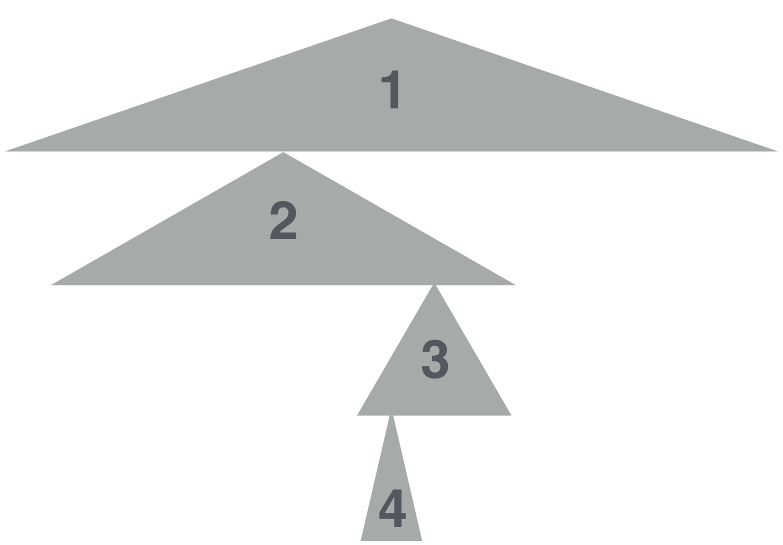


Context-based ID Remapping

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*

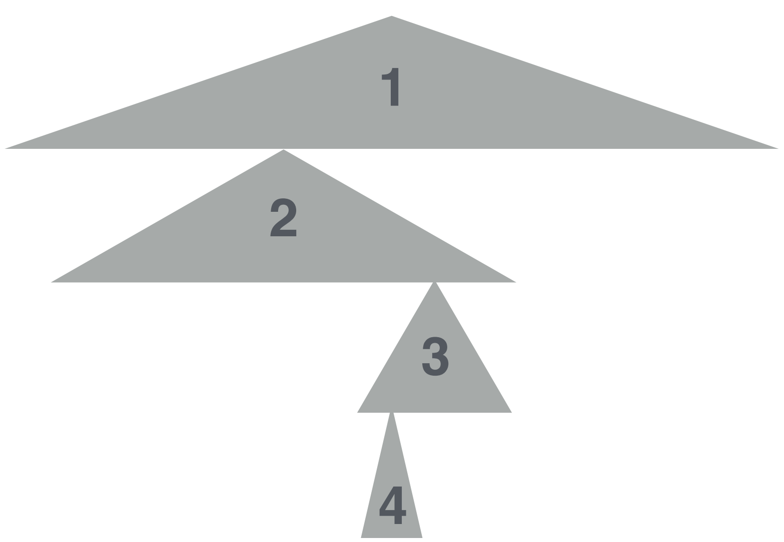
	k	3-grams	4-grams	5-grams
Europarl	0	2404	2782	2920
	1	213 (×11.28)	480 (×5.79)	646 (×4.52)
	2	2404	48 (×57.95)	101 (×28.91)
YahooV2	0	7350	7197	7417
	1	753 (×9.76)	1461 (×4.93)	1963 (×3.78)
	2	7350	104 (×69.20)	249 (×29.79)
GoogleV2	0	4050	6631	6793
	1	1025 (×3.95)	2192 (×3.03)	2772 (×2.45)
	2	4050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

Context-based ID Remapping

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)

Experimental Analysis - EF/PEF (R)Trie

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

Experimental Analysis - EF/PEF (R)Trie

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

Experimental Analysis - EF/PEF (R)Trie

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%)	—	—

Experimental Analysis - EF/PEF (R)Trie

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	μs × query	bpg	μs × query	bpg	μs × 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%)	—	—

Experimental Analysis - EF/PEF (R)Trie

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	μs × query	bpg	μs × query	bpg	μs × 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**!**

Experimental Analysis - EF/PEF (R)Trie

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**!**

Experimental Analysis - EF/PEF (R)Trie

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	μs × query	bpg	μs × query	bpg	μs × 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**?**

Experimental Analysis - Overall comparison

	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%)

Experimental Analysis - Overall comparison

	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%)

Experimental Analysis - Overall comparison

	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%)

Experimental Analysis - Overall comparison

	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%)

Experimental Analysis - Overall comparison

	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]) (+328.87%) 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 (+[illegible]) (+[illegible]) 2.8X	2.06 (+52.00%) (+25.67%)	3.81 (+[illegible]) (+[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

Experimental Analysis - Overall comparison

	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]) (+70.82%) 2X	2.24 (+17.36%) (+61.68%)	9.23 ([illegible]) (+328.87%) 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

Experimental Analysis - Overall comparison

	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 (+[illegible]8%) (+72.70%) 2X	1.69 (−11.41%) (+22.04%)	3.48 (+[illegible]4%) (+61.70%) 2X	1.45 (−4.87%) (+10.83%)	4.13 (+1[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 (+1[illegible]1%) (+70.82%) 2X	2.24 (+17.36%) (+61.68%)	9.23 ([illegible]) (+328.87%) 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 (+3[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.

Experimental Analysis - Perplexity

Reversed Elias-Fano Tries

- Probabilities and backoffs are **quantized** (*binning* method) using any number of bits from 2 to 32
- **Stateful** scoring function

	Europarl				YahooV2			
	bpg		μs × query		bpg		μs × query	
PEF-Trie	3.48		0.25		3.64		0.38	
PEF-RTrie	2.91		0.28		3.06		0.43	
BerkeleyLM C.	6.50	(+87.03%) (+123.47%)	1.19	(+371.79%) (+322.22%)	6.39	(+75.72%) (+109.21%)	1.08	(+187.45%) (+152.17%)
BerkeleyLM H.3	9.36	(+169.17%) (+221.61%)	0.84	(+233.63%) (+198.58%)	8.75	(+140.41%) (+186.23%)	0.74	(+95.77%) (+71.75%)
BerkeleyLM H.50	12.31	(+254.00%) (+322.97%)	0.35	(+39.00%) (+24.39%)	12.01	(+230.05%) (+292.95%)	0.30	(−19.39%) (−29.28%)
Expgram	4.15	(+19.33%) (+42.59%)	3.83	(+1424.87%) (+1264.67%)	5.80	(+59.41%) (+89.79%)	14.05	(+3637.90%) (+3179.16%)
KenLM T.	4.58	(+31.80%) (+57.48%)	0.23	(−8.00%) (−17.66%)	5.04	(+38.53%) (+64.93%)	0.39	(+4.57%) (−8.26%)
RandLM	4.01	(+15.42%) (+37.90%)	6.48	(+2477.95%) (+2207.12%)	3.86	(+6.03%) (+26.24%)	6.25	(+1561.20%) (+1357.33%)

- Elias-Fano Tries substantially **outperform ALL** previous solutions in **both space and time**.
- As fast as the state-of-the-art (KenLM) but up to 65% more space-efficient.

Experimental Analysis - Perplexity

Reversed Elias-Fano Tries

- Probabilities and backoffs are **quantized** (*binning* method) using any number of bits from 2 to 32
- **Stateful** scoring function

	Europarl		YahooV2	
	bpg	μs $\times$ query	bpg	μs $\times$ query
PEF-Trie	3.48	0.25	3.64	0.38
PEF-RTrie	2.91	0.28	3.06	0.43
BerkeleyLM C.	6.50 (+87.03%) (+123.47%)	1.19 (+371.79%) (+322.22%)	6.39 (+75.72%) (+109.21%)	1.08 (+187.45%) (+152.17%)
BerkeleyLM H.3	9.36 (+169.17%) (+221.61%)	0.84 (+233.63%) (+198.58%)	8.75 (+140.41%) (+186.23%)	0.74 (+95.77%) (+71.75%)
BerkeleyLM H.50	12.31 (+254.00%) (+322.97%)	0.35 (+39.00%) (+24.39%)	12.01 (+230.05%) (+292.95%)	0.30 (−19.39%) (−29.28%)
Expgram	4.15 (+19.33%) (+42.59%)	3.83 (+1424.87%) (+1264.67%)	5.80 (+59.41%) (+89.79%)	14.05 (+3637.90%) (+3179.16%)
KenLM T.	4.58 (+31.80%) (+57.48%)	0.23 (−8.00%) (−17.66%)	5.04 (+38.53%) (+64.93%)	0.39 (+4.57%) (−8.26%)
RandLM	4.01 (+15.42%) (+37.90%)	6.48 (+2477.95%) (+2207.12%)	3.86 (+6.03%) (+26.24%)	6.25 (+1561.20%) (+1357.33%)

- Elias-Fano Tries substantially **outperform ALL** previous solutions in **both space and time**.
- As fast as the state-of-the-art (KenLM) but up to 65% more space-efficient.

Experimental Analysis - Perplexity

Reversed Elias-Fano Tries

- Probabilities and backoffs are **quantized** (*binning* method) using any number of bits from 2 to 32
- **Stateful** scoring function

	Europarl		YahooV2	
	bpg	μs × query	bpg	μs × query
PEF-Trie	3.48	0.25	3.64	0.38
PEF-RTrie	2.91	0.28	3.06	0.43
BerkeleyLM C.	6.50 (+87.03%) (+123.47%)	1.19 (+371.79%) (+322.22%)	6.39 (+75.72%) (+109.21%)	1.08 (+187.45%) (+152.17%)
BerkeleyLM H.3	9.36 (+169.17%) (+221.61%)	0.84 (+233.63%) (+198.58%)	8.75 (+140.41%) (+186.23%)	0.74 (+95.77%) (+71.75%)
BerkeleyLM H.50	12.31 (+254.00%) (+322.97%)	0.35 (+39.00%) (+24.39%)	12.01 (+230.05%) (+292.95%)	0.30 (−19.39%) (−29.28%)
Expgram	4.15 (+19.33%) (+42.59%)	3.83 (+1424.87%) (+1264.67%)	5.80 (+59.41%) (+89.79%)	14.05 (+3637.90%) (+3179.16%)
KenLM T.	4.58 [illegible] **+60%** (+57.48%)	0.23 (−8.00%) (−17.66%)	5.04 [illegible] **+65%** (+64.93%)	0.39 (+4.57%) (−8.26%)
RandLM	4.01 (+15.42%) (+37.90%)	6.48 (+2477.95%) (+2207.12%)	3.86 (+6.03%) (+26.24%)	6.25 (+1561.20%) (+1357.33%)

- Elias-Fano Tries substantially **outperform ALL** previous solutions in **both space and time**.
- As fast as the state-of-the-art (KenLM) but up to 65% more space-efficient.

Experimental Analysis - Perplexity

Reversed Elias-Fano Tries

- Probabilities and backoffs are **quantized** (*binning* method) using any number of bits from 2 to 32
- **Stateful** scoring function

	Europarl bpg	Europarl $\mu s \times$ query	YahooV2 bpg	YahooV2 $\mu s \times$ query
PEF-Trie	3.48	0.25	3.64	0.38
PEF-RTrie	2.91	0.28	3.06	0.43
BerkeleyLM C.	6.50 (+87.03%) (+123.47%)	1.19 (+371.79%) (+322.22%)	6.39 (+75.72%) (+109.21%)	1.08 (+187.45%) (+152.17%)
BerkeleyLM H.3	9.36 (+169.17%) (+221.61%)	0.84 (+233.63%) (+198.58%)	8.75 (+140.41%) (+186.23%)	0.74 (+95.77%) (+71.75%)
BerkeleyLM H.50	12.31 (+254.00%) (+322.97%)	0.35 (+39.00%) (+24.39%)	12.01 (+230.05%) (+292.95%)	0.30 (−19.39%) (−29.28%)
Expgram	4.15 (+19.33%) (+42.59%)	3.83 (+1424.87%) (+1264.67%)	5.80 (+59.41%) (+89.79%)	14.05 (+3637.90%) (+3179.16%)
KenLM T.	4.58 [illegible] (+57.48%) **+60%**	0.23 (−8.00%) (−17.66%)	5.04 [illegible] (+64.93%) **+65%**	0.39 (+4.57%) (−8.26%)
RandLM	4.01 (+15.42%) (+37.90%)	6.48 (+2477.95%) (+2207.12%)	3.86 (+6.03%) (+26.24%)	6.25 (+1561.20%) (+1357.33%)

- Elias-Fano Tries substantially **outperform ALL** previous solutions in **both space and time**.
- As fast as the state-of-the-art (KenLM) but up to 65% more space-efficient.

Experimental Analysis - Perplexity

Reversed Elias-Fano Tries

- Probabilities and backoffs are **quantized** (*binning* method) using any number of bits from 2 to 32
- **Stateful** scoring function

	Europarl		YahooV2	
	bpg	$\mu s \times$ query	bpg	$\mu s \times$ query
PEF-Trie	3.48	0.25	3.64	0.38
PEF-RTrie	2.91	0.28	3.06	0.43
BerkeleyLM C.	6.50 (+87.03%) (+[illegible]%) 2X	1.19 (+[illegible]%) (+322.22%) 4X	6.39 (+75.[illegible]%) (+109.[illegible]%) 2X	1.08 (+[illegible]%) (+152.17%) 2.7X
BerkeleyLM H.3	9.36 (+[illegible]%) (+221.61%) 3X	0.84 ([illegible]%) (+198.58%) 3.3X	8.75 (+[illegible]%) (+186.23%) 2.5X	0.74 (+9[illegible]%) (+71.75%) 2X
BerkeleyLM H.50	12.31 (+[illegible]%) (+322.97%) 4X	0.35 ([illegible]%) (+24.39%) +25%	12.01 (+[illegible]%) (+292.95%) 3.5X	0.30 (−19.39%) (−29.28%)
Expgram	4.15 ([illegible]%) (+42.59%) +40%	3.83 (+[illegible]%) (+1264.67%) 15X	5.80 ([illegible]%) (+89.79%) +80%	14.05 (+[illegible]%) (+3179.16%) 36X
KenLM T.	4.58 ([illegible]%) (+57.48%) +60%	0.23 (−8.00%) (−17.66%)	5.04 ([illegible]%) (+64.93%) +65%	0.39 (+4.57%) (−8.26%)
RandLM	4.01 (+[illegible]%) (+37.90%) +30%	6.48 (+[illegible]%) (+2207.12%) 25X	3.86 (+[illegible]%) (+26.24%) +20%	6.25 (+[illegible]%) (+1357.33%) 16X

- Elias-Fano Tries substantially **outperform ALL** previous solutions in **both space and time**.
- As fast as the state-of-the-art (KenLM) but up to 65% more space-efficient.

https://github.com/jermp/tongrams

jermp / tongrams

Unwatch 1 | Star 6 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Settings | Insights

The C++ library implementing the compressed data structures described in the paper "Efficient Data Structures for Massive N-Gram Datasets", by Giulio Ermanno Pibiri and Rossano Venturini, published in ACM SIGIR 2017.

Edit

trie | elias-fano | ngrams | Manage topics

18 commits | 1 branch | 1 release | 1 contributor

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

jermp added compiler version to README — Latest commit b80e241 on Jun 21

emphf @ a18574f	added emphf submodule	3 months ago
sequences	added new select-in-word algorithm; CMakeLists.txt updated; README.md...	2 months ago
sorters	code imported	3 months ago
test	code imported	3 months ago
test_data	code imported	3 months ago
utils	added new select-in-word algorithm; CMakeLists.txt updated; README.md...	2 months ago
vectors	code imported	3 months ago

https://github.com/jermp/tongrams

`tongrams` - Tons of *N*-Grams

`tongrams` is a C++ library implementing the compressed data structures described in the paper *Efficient Data Structures for Massive N-Gram Datasets*, by Giulio Ermanno Pibiri and Rossano Venturini, published in ACM SIGIR 2017 [1]. The proposed data structures can be used to map *N*-grams to their corresponding (integer) frequency counts or to (floating point) probabilities and backoffs for backoff-interpolated Knenser-Ney models.

The library features a compressed trie data structure in which *N*-grams are assigned integer identifiers (IDs) and compressed with *Elias-Fano* (Subsection 3.1 of [1]) as to support efficient searches within compressed space. The *context-based remapping* of such identifiers (Subsection 3.2 of [1]) permits to encode a word following a context of fixed length *k*, i.e., its preceding *k* words, with an integer whose value is bounded by the number of words that follow such context and *not* by the size of the whole vocabulary (number of uni-grams). Additionally to the trie data structure, the library allows to build models based on *minimal perfect hashing* (MPH), for constant-time retrieval (Section 4 of [1]).

When used to store frequency counts, the data structures support a `lookup()` operation that returns the number of occurrences of the specified *N*-gram. Differently, when used to store probabilities and backoffs, the data structures implement a `score()` function that, given a text as input, computes the perplexity score of the text.

This guide is meant to provide a brief overview of the library and to illustrate its funtionalities through some examples.

Table of contents

https://github.com/jermp/tongrams

`tongrams` - Tons of *N*-Grams

`tongrams` is a C++ library implementing the compressed data structures described in the paper *Efficient Data Structures for Massive N-Gram Datasets*, by Giulio Ermanno Pibiri and Rossano Venturini, published in ACM SIGIR 2017 [1]. The proposed data structures can be used to map *N*-grams to their corresponding (integer) frequency counts or to (floating point) probabilities and backoffs for backoff-interpolated Knenser-Ney models.

The library features a compressed trie data structure in which *N*-grams are assigned integer identifiers (IDs) and compressed with *Elias-Fano* (Subsection 3.1 of [1]) as to support efficient searches within compressed space. The *context-based remapping* of such identifiers (Subsection 3.2
fixed length *k*, i.e., its preceding *k* words, with an integer who
such context and *not* by the size of the whole vocabulary (nu
structure, the library allows to build models based on *minima*
(Section 4 of [1]).

When used to store frequency counts, the data structures su
occurrences of the specified *N*-gram. Differently, when used
implement a `score()` function that, given a text as input, co

This guide is meant to provide a brief overview of the library and to illustrate its funtionalities through some examples.

Table of contents

On-going work

- Parallel and scalable estimation of *Kneser-Ney* language models
- `Python` wrapper, installable through `pip` utility

https://github.com/jermp/tongrams

`tongrams` - Tons of *N*-Grams

`tongrams` is a C++ library implementing the compressed data structures described in the paper *Efficient Data Structures for Massive N-Gram Datasets*, by Giulio Ermanno Pibiri and Rossano Venturini, published in ACM SIGIR 2017 [1]. The proposed data structures can be used to map *N*-grams to their corresponding (integer) frequency counts or to (floating point) probabilities and backoffs for backoff-interpolated Knenser-Ney models.

The library features a compressed trie data structure in which *N*-grams are assigned integer identifiers (IDs) and compressed with *Elias-Fano* (Subsection 3.1 of [1]) as to support efficient searches within compressed space. The *context-based remapping* of such identifiers (Subsection 3.2
fixed length *k*, i.e., its preceding *k* words, with an integer who
such context and *not* by the size of the whole vocabulary (nu
structure, the library allows to build models based on *minima*
(Section 4 of [1]).

When used to store frequency counts, the data structures su
occurrences of the specified *N*-gram. Differently, when used
implement a `score()` function that, given a text as input, co

This guide is meant to provide a brief overview of the library and to illustrate its funtionalities through some examples.

Table of contents

On-going work

- Parallel and scalable estimation of *Kneser-Ney* language models
- `Python` wrapper, installable through `pip` utility

Future work

- Optimal ID-assignment for Elias-Fano? (NP-hard problem)
- Make queries (especially perplexity) even faster

Proudly supported by a

SIGIR Special Interest Group on Information Retrieval **Student Travel Grant**

Thanks for your attention, time, patience!

Any questions?