CS6203: Advanced Topics in Database Management Systems

The Case for Learned Index Structures

HUANG liu

e0575772@u.nus.edu

Introduction

Range Index

Point Index

Existence Index

Related Work & Conclusion

1 Introduction

Index structures

DATA ACCESS PATTERNS

range request (from sorted array) → B-tree

single key look-ups (from unsorted array) → HashMap

check for record existence → Bloom Filter

✓ General cases

☹ Specific case

Array: [100, 101, … 100M]

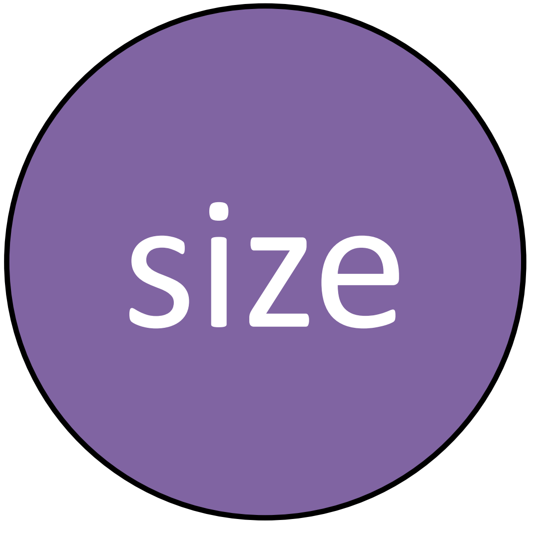
B-Tree: O(log(n))

Array[key - 100]: O(1)

With the knowledge of data distribution, we can highly optimize the index structure.

Traditional data structures

Data structures using ML to learn from data



Learned Index

ML + **Tradition**

ML **~~Tradition~~**

2 Range Index

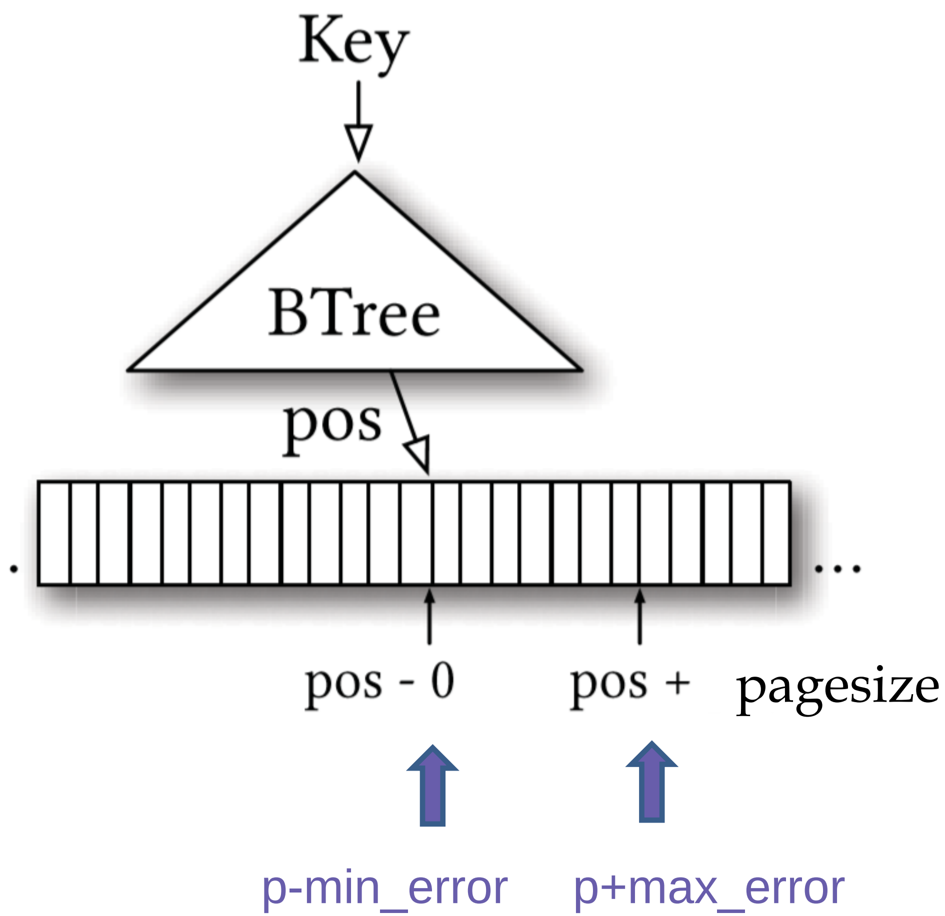
B-Tree is a model

Assumptions:

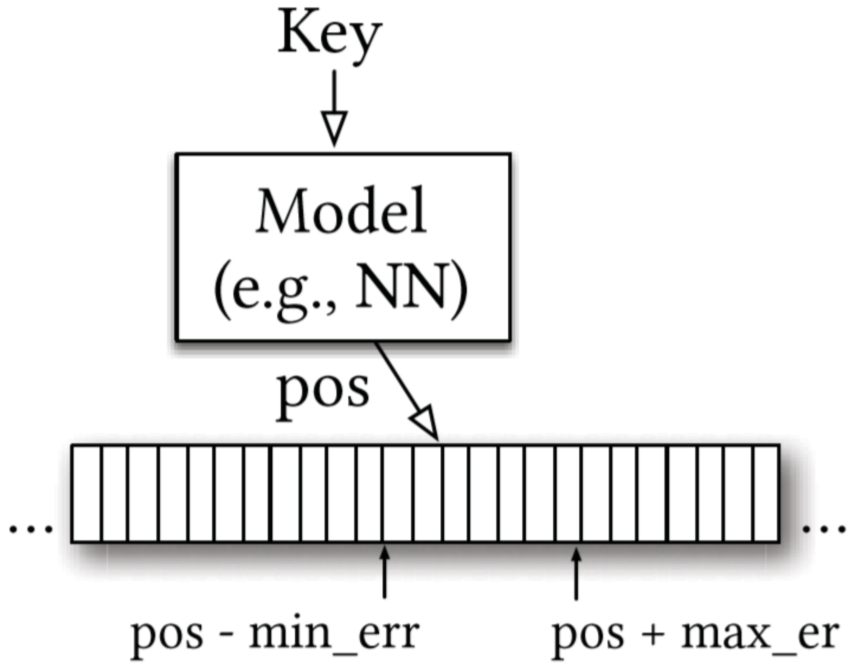
In-memory database

(i.e., read-only)

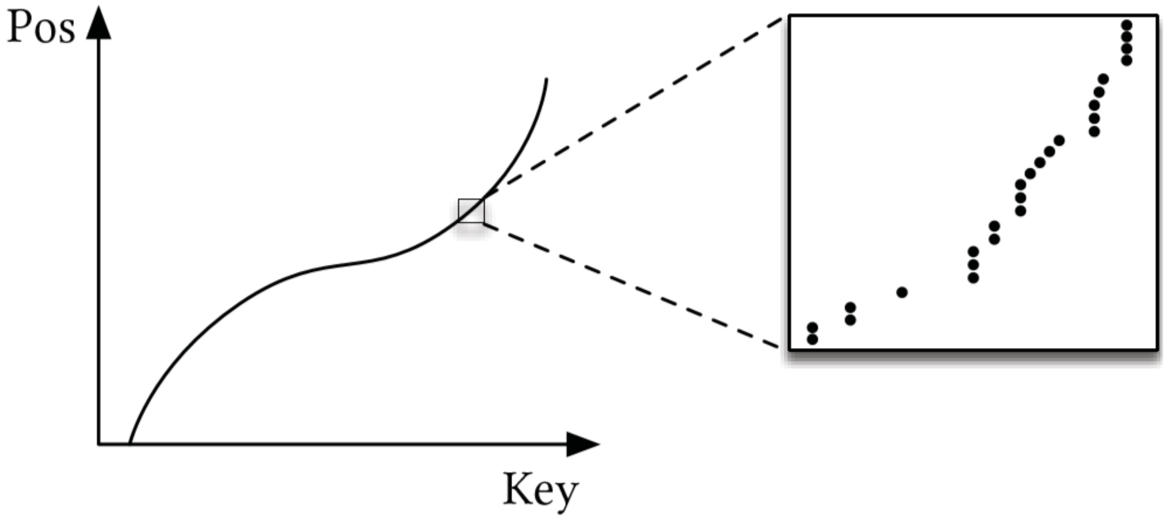
No insert



Another view of index structure



Min_err and Max_err are known from the training process.
Eg. For B-trees, the error is the page_size.



$$pos = F(key) * \#Keys$$
$$F(key) = P(x \leq key)$$

- A model to predict the position of a given key is actually building a model to present the CDF(Cumulative Distribution Function) of data.

B-tree: regression tree

$\Downarrow$

ML models: linear regression, NN…

Benefits of learned index models

- Smaller index: less main-memory storage.
- Faster lookup: eg. Linear function.
- More parallelism: use multiplications instead of if-statement.

Benefits of learned index depend on:

- How accurately the model represents the observed CDF
- Architecture, implementation details, payload

A First Naive Learned Index

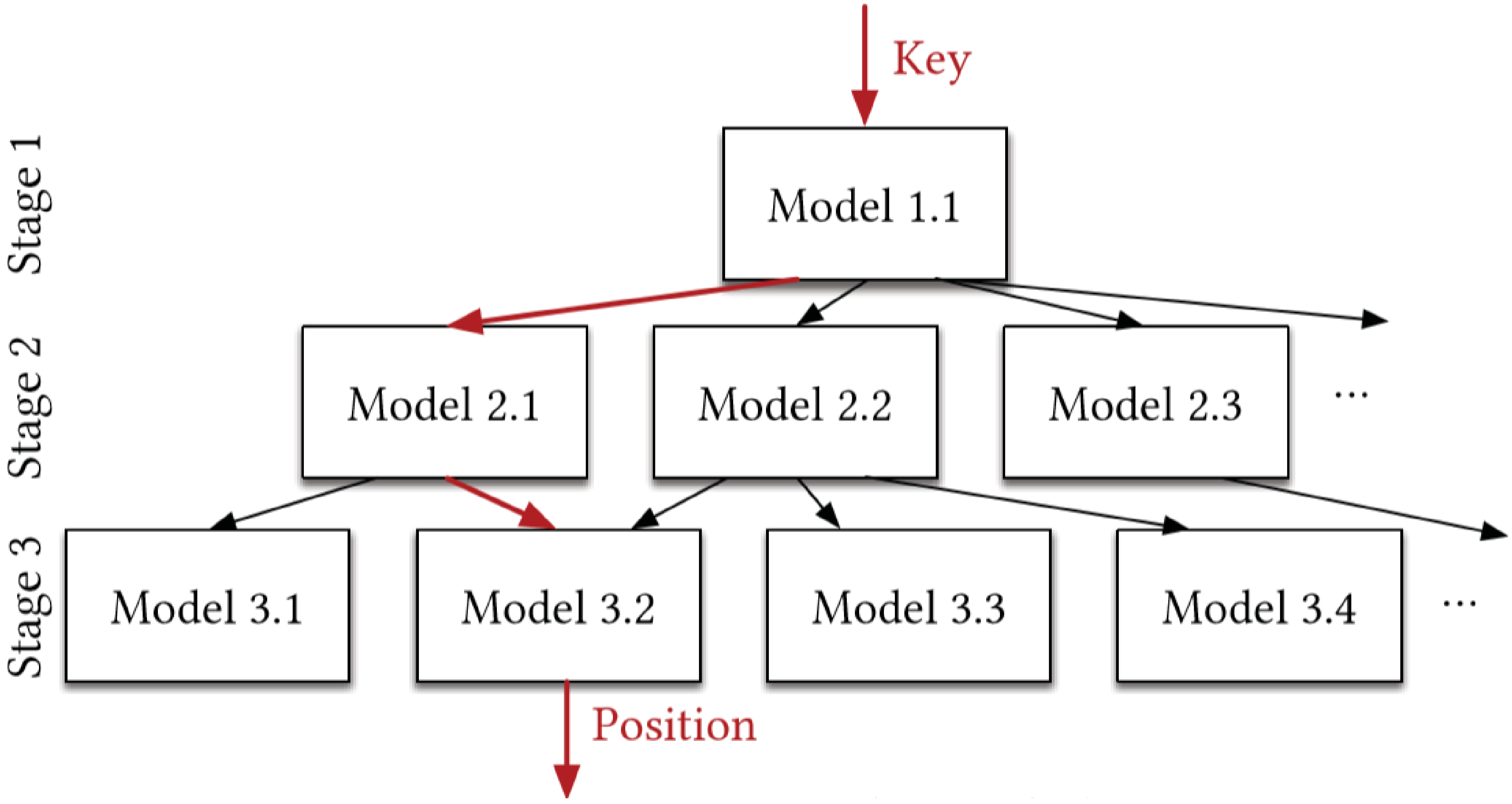
- Tensorflow
- 200M web-server log records
- Input features: timestamps
- Labels: positions in sorted array
- 2 fc , 32 neurons/layer, ReLu activated

300ns 80000ns

REASONS:

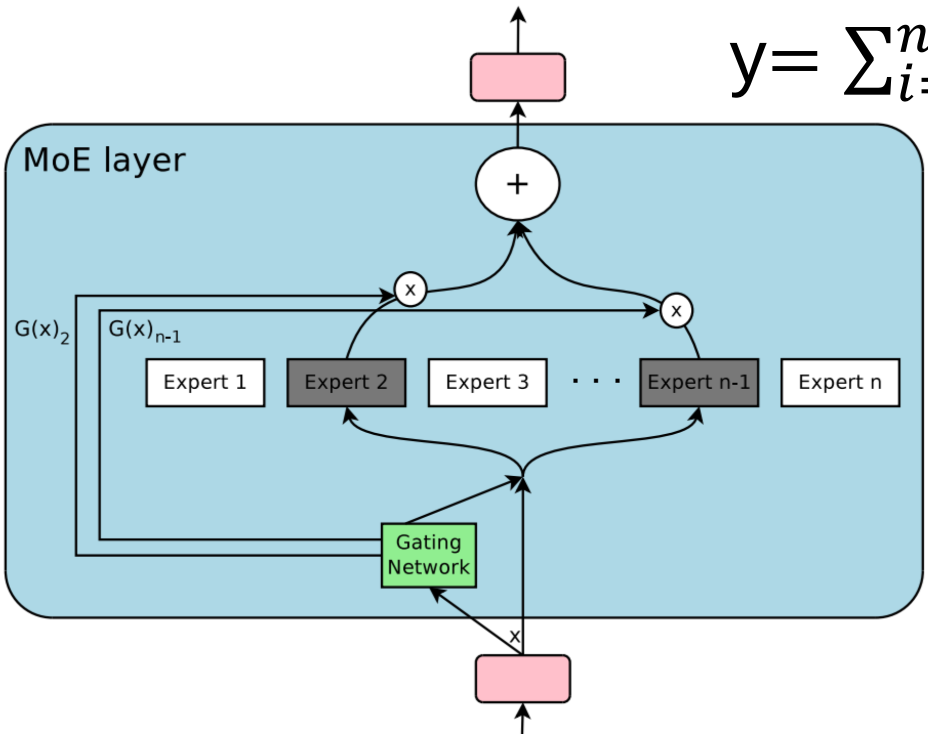
1. Tensorflow is designed for large models
2. B-tree can easily overfit the training data and can predict the position precisely
3. B-trees are cache efficient

Solution: Recursive Model Index



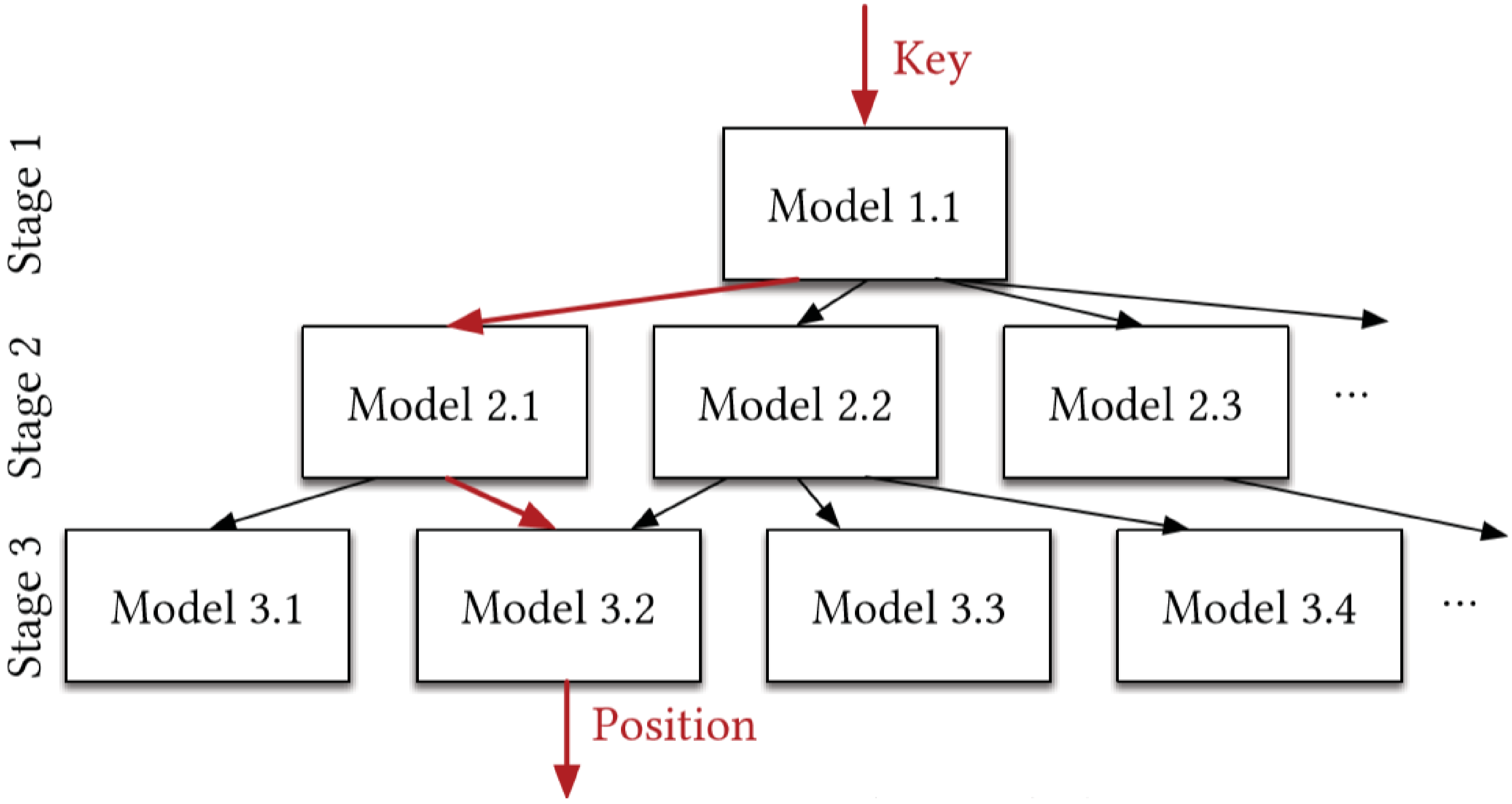
A Mixture of Experts (MoE) layer

$$y= \sum_{i=1}^{n} G(x)iEi(x)$$



N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixtureof-experts layer. arXiv preprint arXiv:1701.06538, 2017.

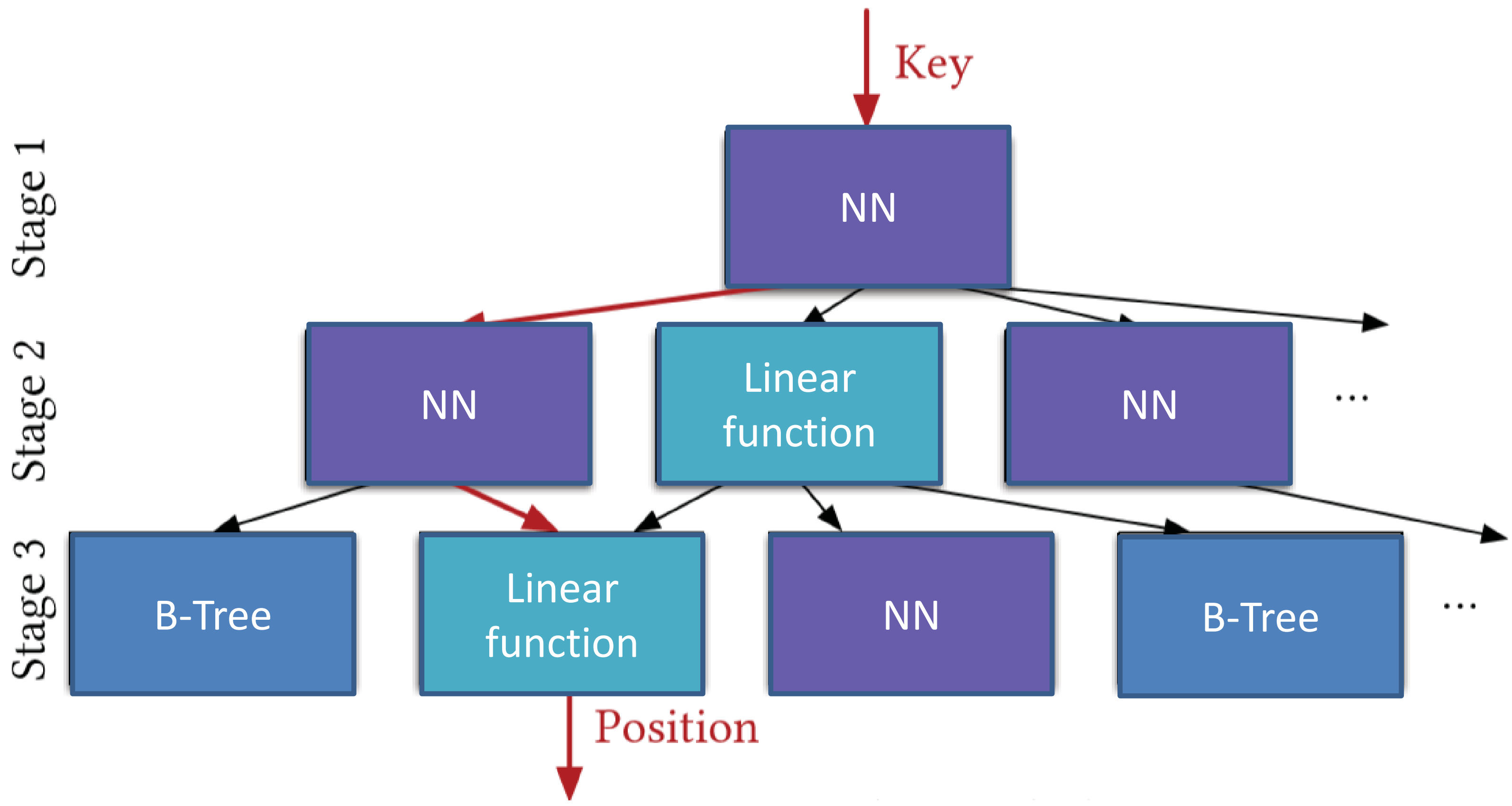
Solution: Recursive Model Index



Advantages of RMI

- separates model size and complexity from execution cost
- easy to learn the overall data distribution
- easier to solve the last-mile accuracy problem.
- no search process is required in-between stages.

Hybrid Indexes

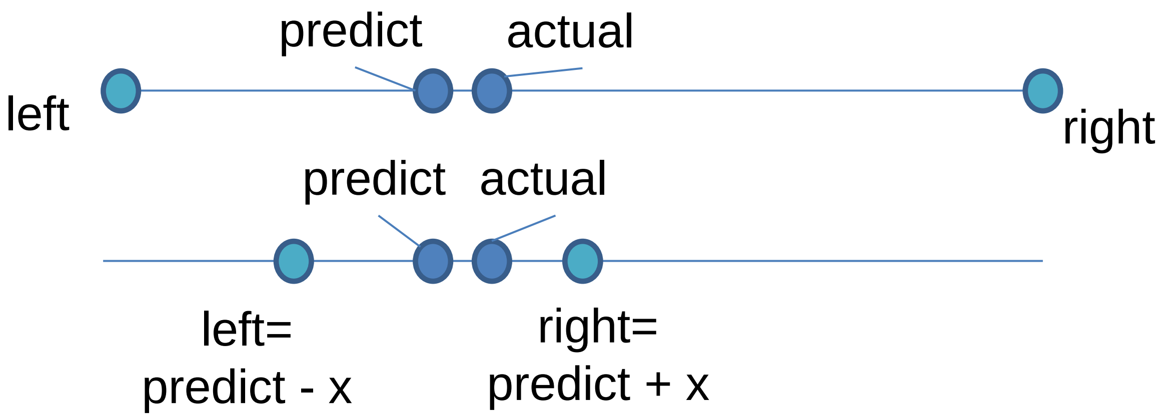


Search Strategies

Binary Search

https://images.app.goo.gl/P7mvfhhx1vaXCKA69

Quaternary Search



Indexing Strings

- Tokenization: $x = (x1, x2,\ldots, xn)$
- xi is the ASCII decimal value of the corresponding character.
- Set a fixed length N: For strings with length $n<N$, we set $xi = 0$ for $i > n$.

Result

		Map Data			Web Data			Log-Normal Data		
Type	**Config**	Size (MB)	Lookup (ns)	Model (ns)	Size (MB)	Lookup (ns)	Model (ns)	Size (MB)	Lookup (ns)	Model (ns)
Btree	page size: 32	52.45 (4.00x)	274 (0.97x)	198 (72.3%)	51.93 (4.00x)	276 (0.94x)	201 (72.7%)	49.83 (4.00x)	274 (0.96x)	198 (72.1%)
	page size: 64	26.23 (2.00x)	277 (0.96x)	172 (62.0%)	25.97 (2.00x)	274 (0.95x)	171 (62.4%)	24.92 (2.00x)	274 (0.96x)	169 (61.7%)
	page size: 128	13.11 (1.00x)	265 (1.00x)	134 (50.8%)	12.98 (1.00x)	260 (1.00x)	132 (50.8%)	12.46 (1.00x)	263 (1.00x)	131 (50.0%)
	page size: 256	6.56 (0.50x)	267 (0.99x)	114 (42.7%)	6.49 (0.50x)	266 (0.98x)	114 (42.9%)	6.23 (0.50x)	271 (0.97x)	117 (43.2%)
	page size: 512	3.28 (0.25x)	286 (0.93x)	101 (35.3%)	3.25 (0.25x)	291 (0.89x)	100 (34.3%)	3.11 (0.25x)	293 (0.90x)	101 (34.5%)
Learned Index	2nd stage models: 10k	0.15 (0.01x)	98 (2.70x)	31 (31.6%)	0.15 (0.01x)	222 (1.17x)	29 (13.1%)	0.15 (0.01x)	178 (1.47x)	26 (14.6%)
	2nd stage models: 50k	0.76 (0.06x)	85 (3.11x)	39 (45.9%)	0.76 (0.06x)	162 (1.60x)	36 (22.2%)	0.76 (0.06x)	162 (1.62x)	35 (21.6%)
	2nd stage models: 100k	1.53 (0.12x)	82 (3.21x)	41 (50.2%)	1.53 (0.12x)	144 (1.81x)	39 (26.9%)	1.53 (0.12x)	152 (1.73x)	36 (23.7%)
	2nd stage models: 200k	3.05 (0.23x)	86 (3.08x)	50 (58.1%)	3.05 (0.24x)	126 (2.07x)	41 (32.5%)	3.05 (0.24x)	146 (1.79x)	40 (27.6%)

- Baseline: B-tree, dense pages
- RMI: 2-stage, NN with 0-2 hidden layers, layer width 4-32

Other baseline

	Lookup Table w/ AVX search	FAST	Fixe-Size Btree w/ interpol. search	Multivariate Learned Index
Time	199 ns	189 ns	280 ns	105 ns
Size	16.3 MB	1024 MB	1.5 MB	1.5 MB

Log normal data with a payload of an eight-byte pointer

Index over String

	Config	Size(MB)	Lookup (ns)	Model (ns)
Btree	page size: 32	13.11 (4.00x)	1247 (1.03x)	643 (52%)
	page size: 64	6.56 (2.00x)	1280 (1.01x)	500 (39%)
	page size: 128	3.28 (1.00x)	1288 (1.00x)	377 (29%)
	page size: 256	1.64 (0.50x)	1398 (0.92x)	330 (24%)
Learned Index	1 hidden layer	1.22 (0.37x)	1605 (0.80x)	503 (31%)
	2 hidden layers	2.26 (0.69x)	1660 (0.78x)	598 (36%)
Hybrid Index	t=128, 1 hidden layer	1.67 (0.51x)	1397 (0.92x)	472 (34%)
	t=128, 2 hidden layers	2.33 (0.71x)	1620 (0.80x)	591 (36%)
	t= 64, 1 hidden layer	2.50 (0.76x)	1220 (1.06x)	440 (36%)
	t= 64, 2 hidden layers	2.79 (0.85x)	1447 (0.89x)	556 (38%)
Learned QS	1 hidden layer	1.22 (0.37x)	1155 (1.12x)	496 (43%)

Non-hybrid RMI with quaternary search

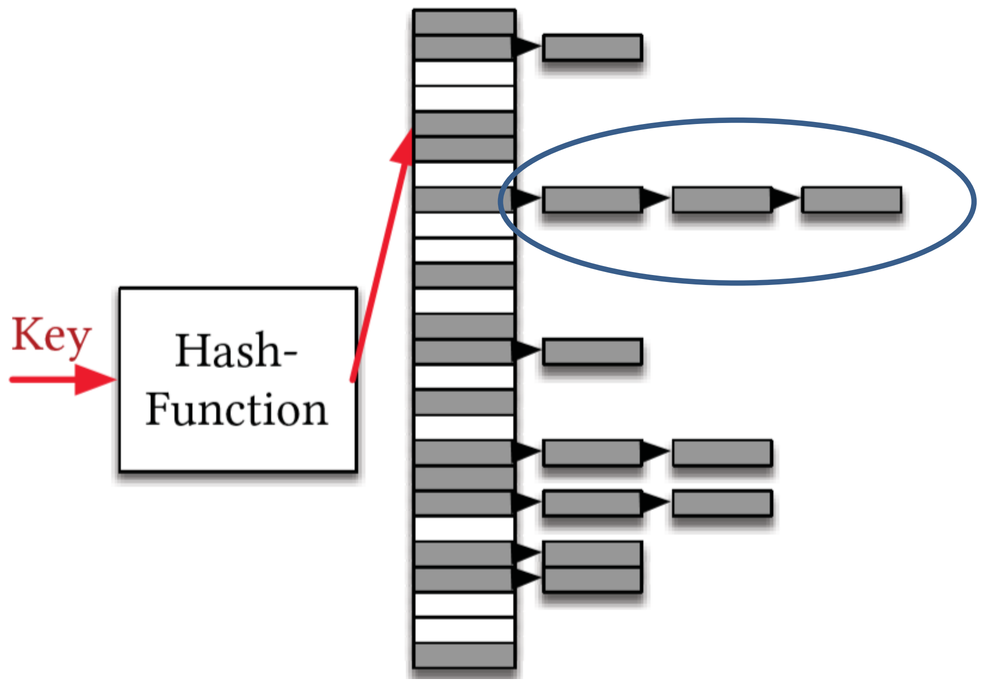
All RMI indexes used 10000 models on the 2nd stage

3 Point Index

Point Index

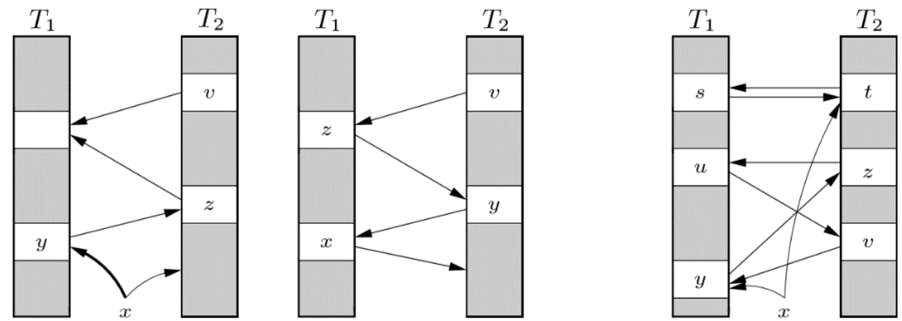
Point Index is the structure that maps keys to positions inside an array, i.e., Hash-Map

(a) Traditional Hash-Map



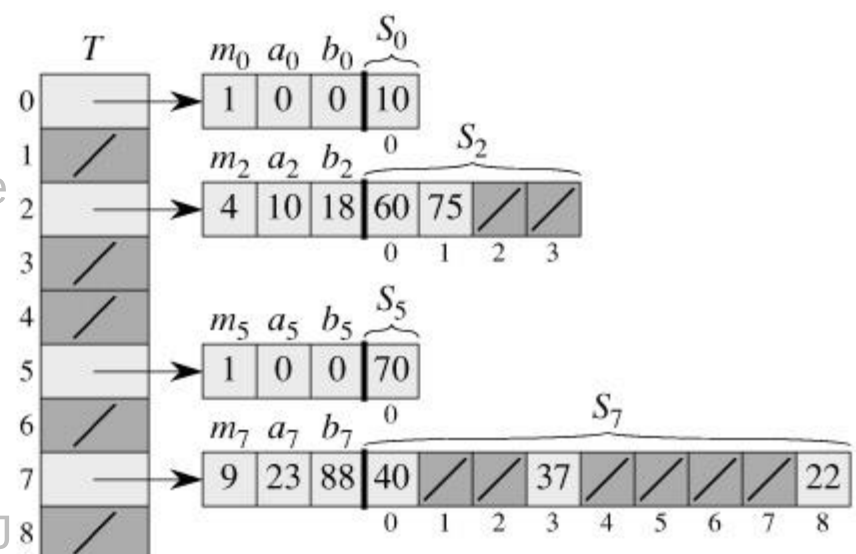
Reduce Conflicts

- More than one hashing



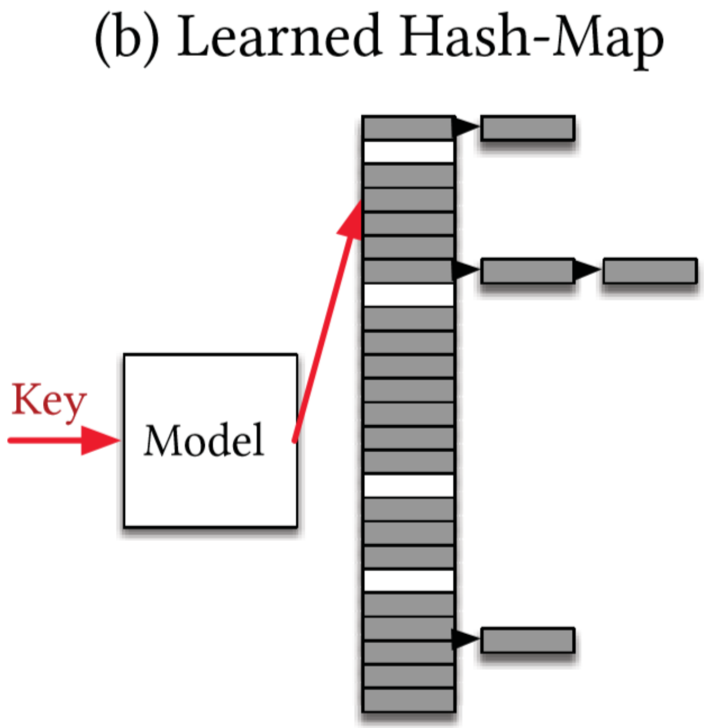
R. Pagh and F. F. Rodler. Cuckoo hashing. Journal of Algorithms, 51(2):122– 144, 2004.

- Perfect hashing



M.Dietzfelbinger,A.Karlin,K.Mehlhorn,F.MeyerauFderHeide,H.Rohnert, and R. E. Tarjan. Dynamic perfect hashing: Upper and lower bounds. SIAM Journal on Computing, 23(4):738–761, 1994.
Image from: https://images.app.goo.gl/Sk7ADonEwLi6JE6U7

Learned hash function



- $h(k) = F(k) * M$
- M: target size
- Use the function F to learn the empirical CDF of the keys
- Can be combined with chaining or other hash-map type

Result

	% Conflicts Hash Map	% Conflicts Model	Reduction
Map Data	35.3%	07.9%	77.5%
Web Data	35.3%	24.7%	30.0%
Log Normal	35.4%	25.9%	26.7%

Figure 8: Reduction of Conflicts

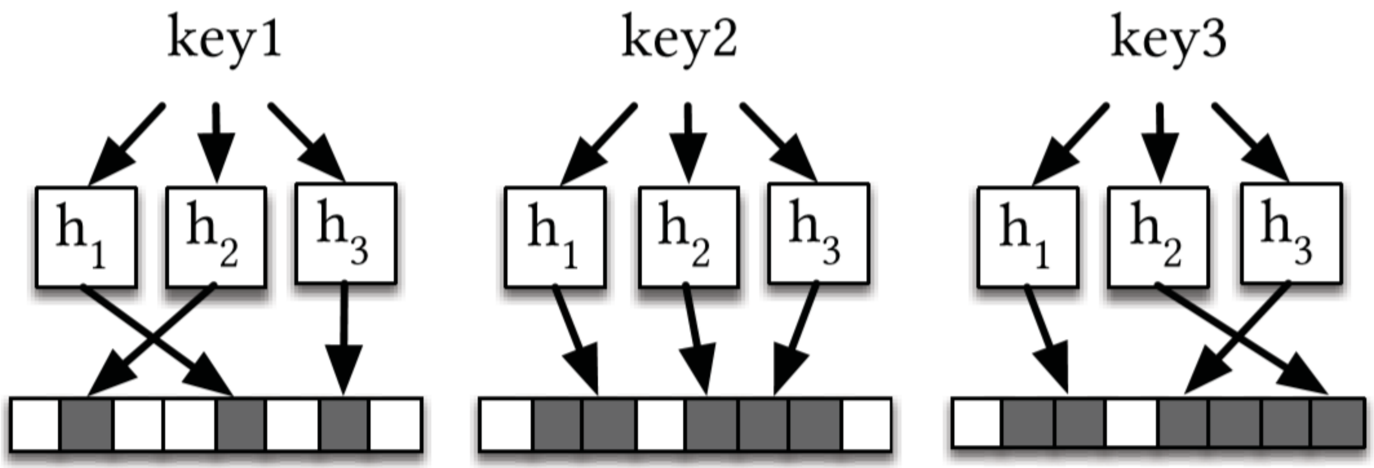
- Model: 2-stage RMI models with 100k models on the 2nd stage, no hidden layers.
- Baseline: MurmurHash3-like hash-function.

Fewer conflicts lead to fewer cache misses and better performance.

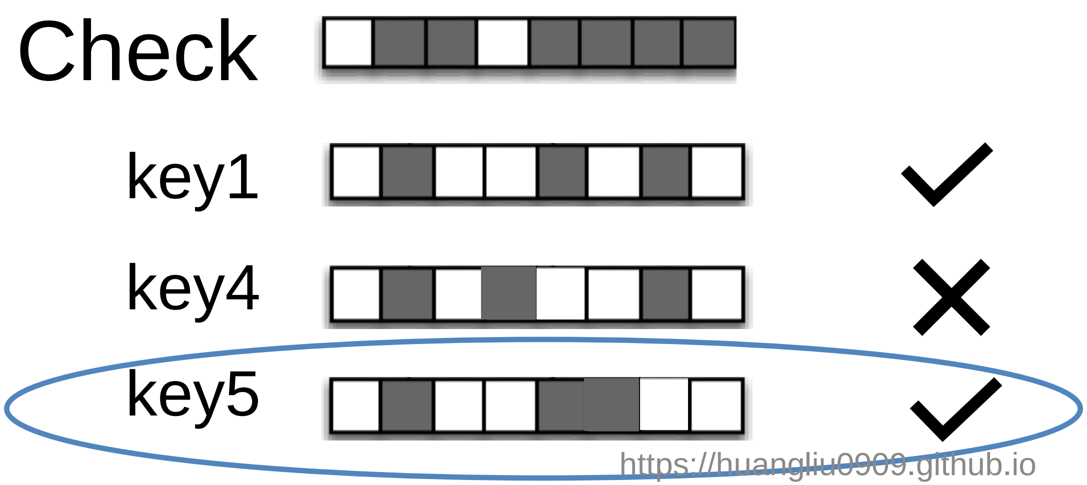
4 Existence Index

Traditional Bloom-Filter

Insert



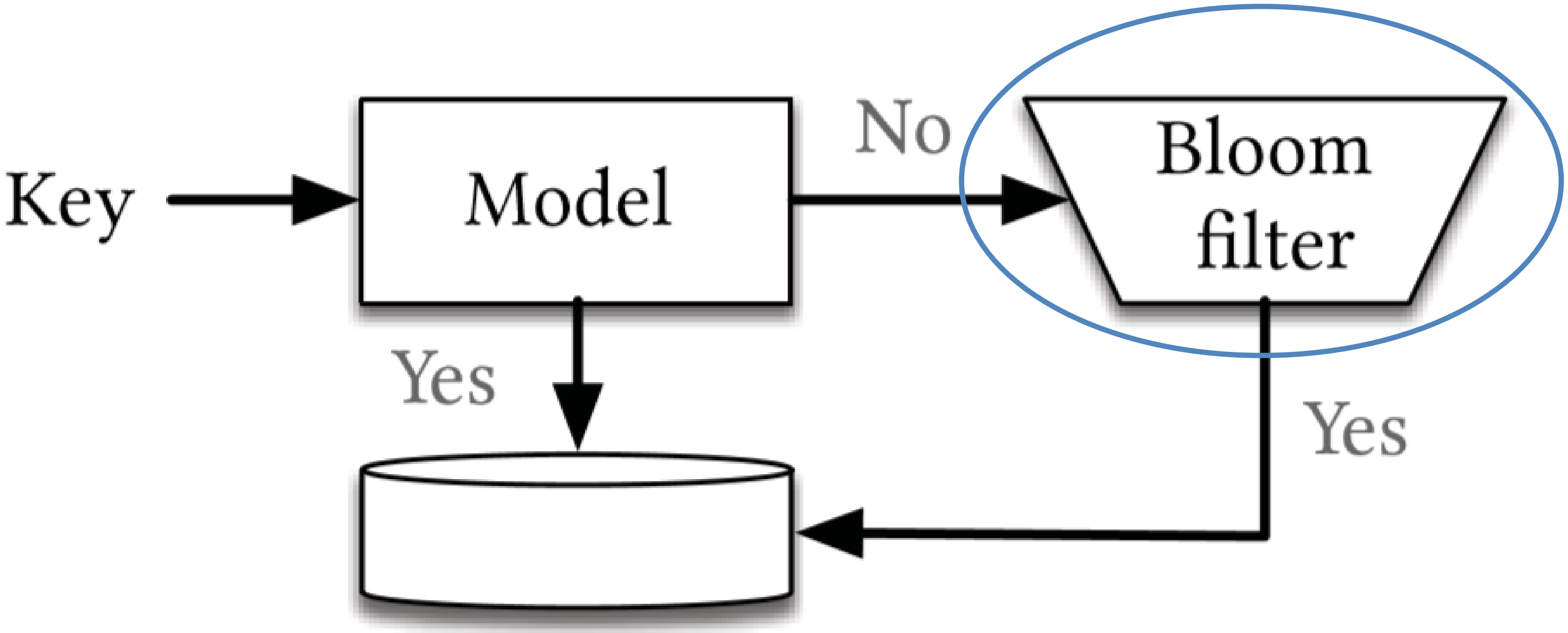
Check



No False Negative

False Positive !

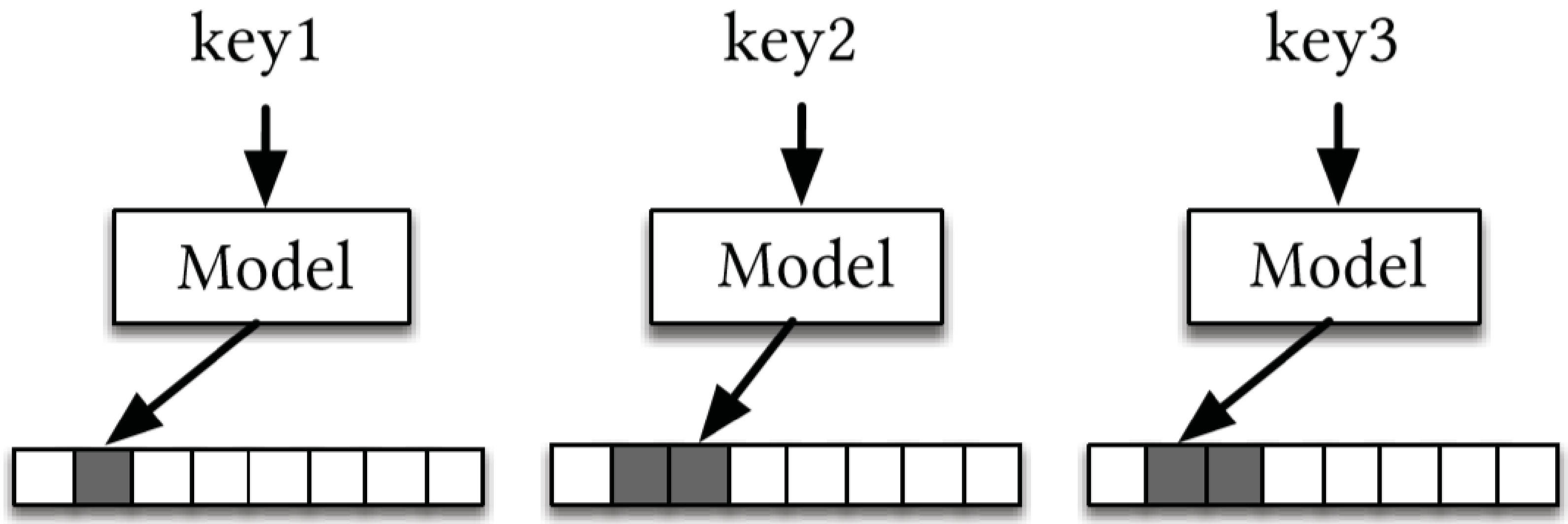
Learned Bloom filters as a classification problem



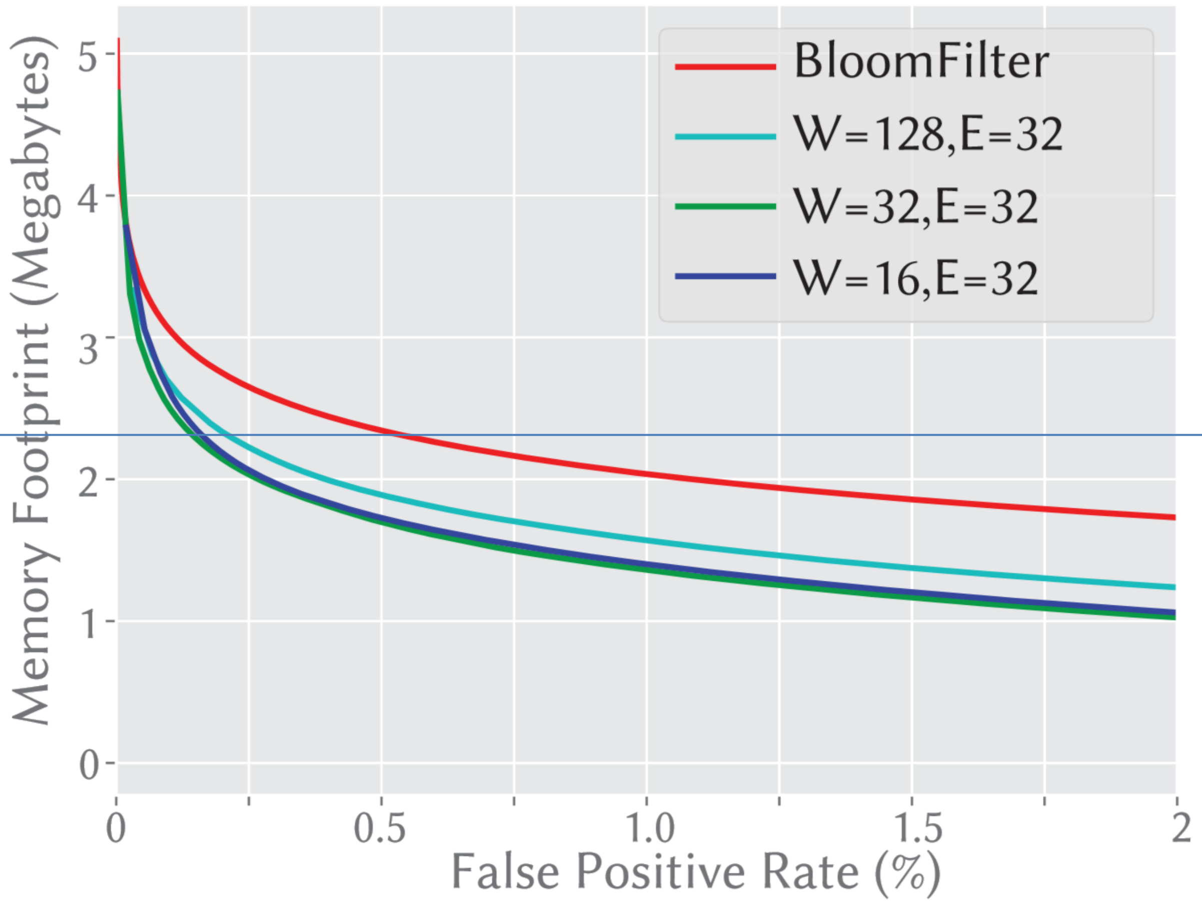
Goal: reduce False Positive Rate

Learned Bloom filters with Model-Hashes

Map most keys to higher range of bit positions and non-keys to the lower range.



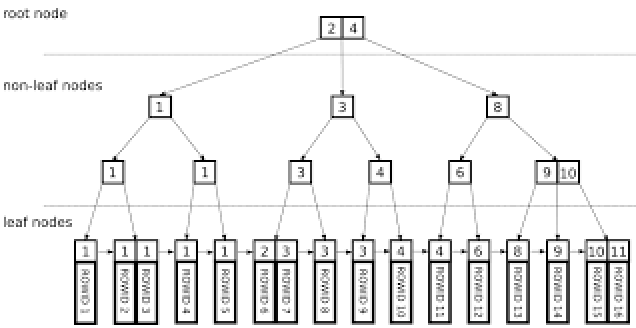
Results



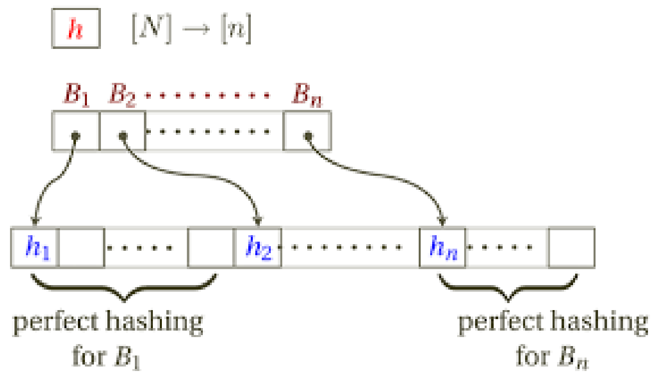
W: neuron number per layer of RNN
E: embedding size for each character

5

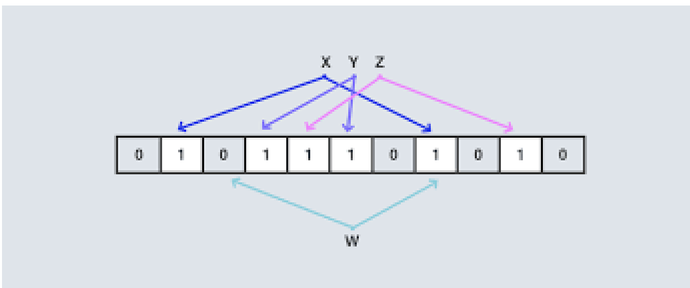
Related Work & Conclusion



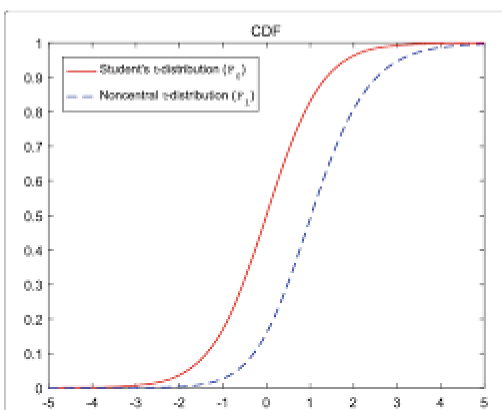
https://images.app.goo.gl/5vU6ZDVxZYfLAchh9



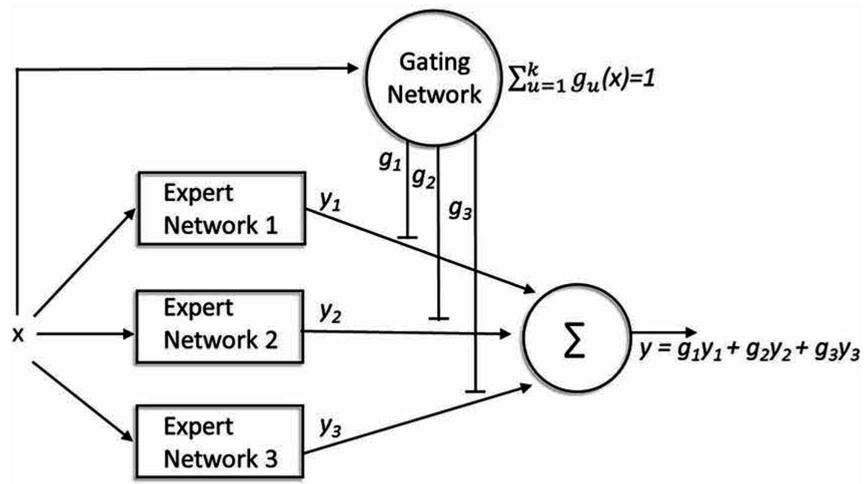
http://tcs.nju.edu.cn/wiki/index.php/%E9%9A%8F%E6%9C%BA%E7%AE%97%E6%B3%95_%28Fall_2011%29/Perfect_hashing



https://images.app.goo.gl/3FFo7W8xXoD87XYdA



https://images.app.goo.gl/zZpd8BnccVutcMx66



https://images.app.goo.gl/m199gdAeoX94X7Xv8

Future Work

- Other ML Models
- Multi-Dimensional Indexes
- Learned Algorithms beyond indexing
- GPU/TPU

“In summary, we have demonstrated that machine learned models have the potential to provide significant benefits over state-of-the-art indexes, and we believe this is a fruitful direction for future research.”

THANKS