The Case for Learned Index Structures

Kraska, T., Beutel, A., Chi, E. H., Dean, J., & Polyzotis, N.

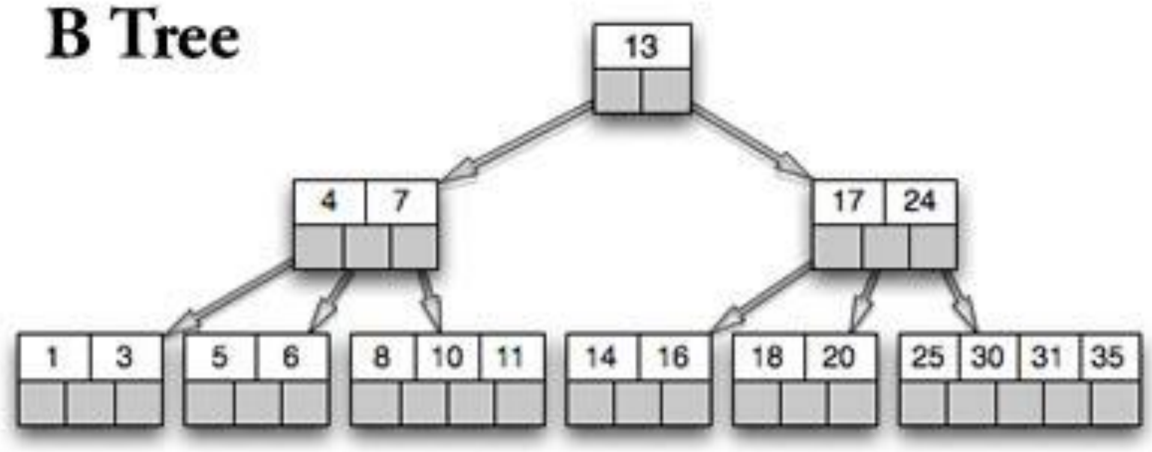
R244

Michael Chi Ian Tang

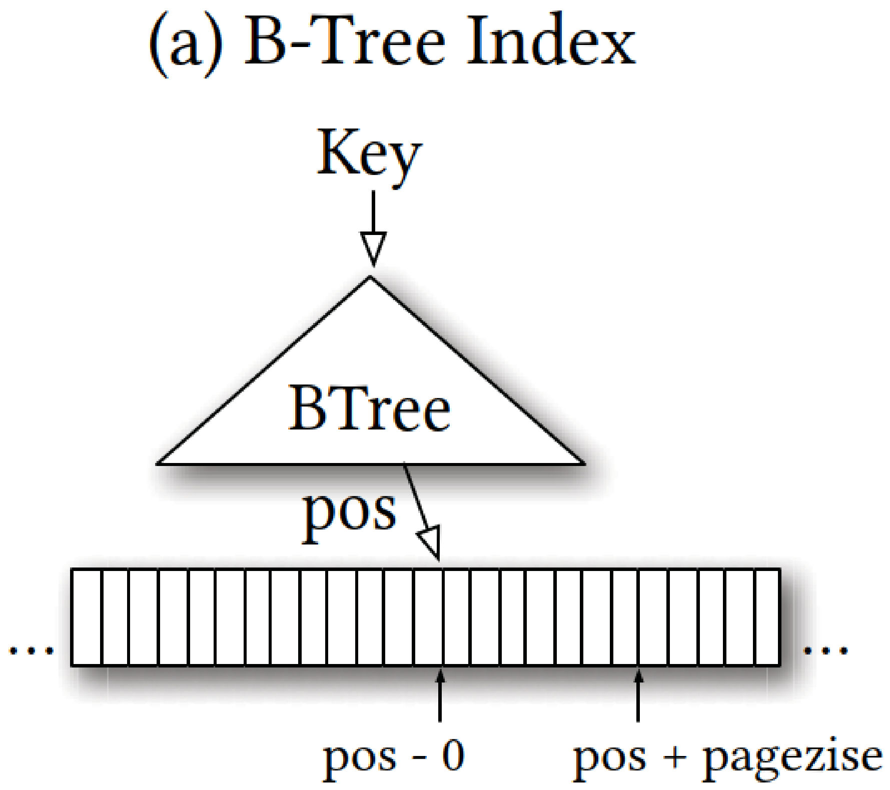
Background

Index Structures

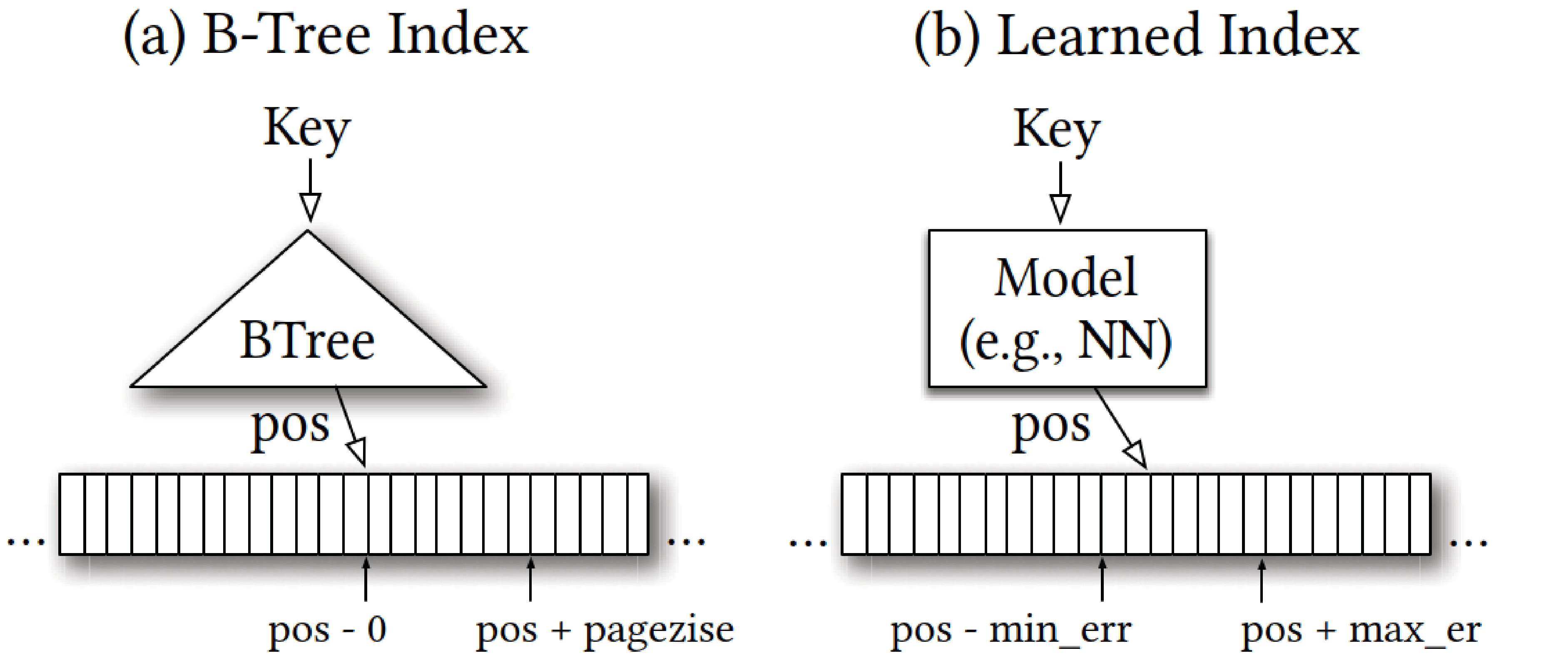
- Index structures are built for efficient data access
- E.g. B-Trees



Index Structures as Models

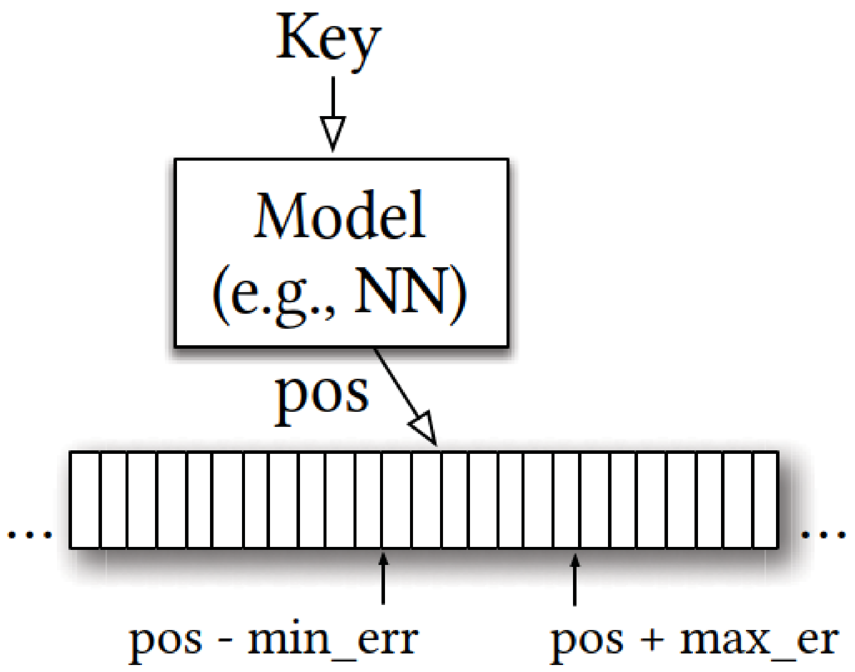


Index Structures as Models



Range Index

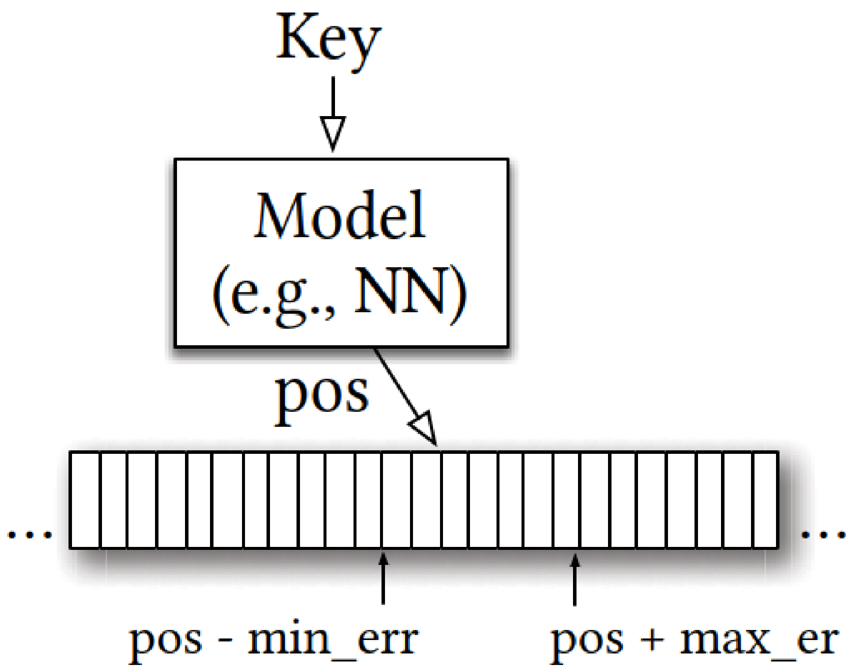
Range Index Models = CDF Models



True position $p*$
$= \ rank(key)$
$= |\{k|k \leq key\}|$
$= P(X \leq key) * N$

$P(X \leq key)$ is the CDF of keys

Range Index Models = CDF Models



Model:

$$pos = F(key) * N \approx p^*$$

$$F(key) \approx P(X \leq key)$$

The Recursive Model Index (RMI)

- Prediction from previous stage chooses the next model
- Progressively refine the prediction

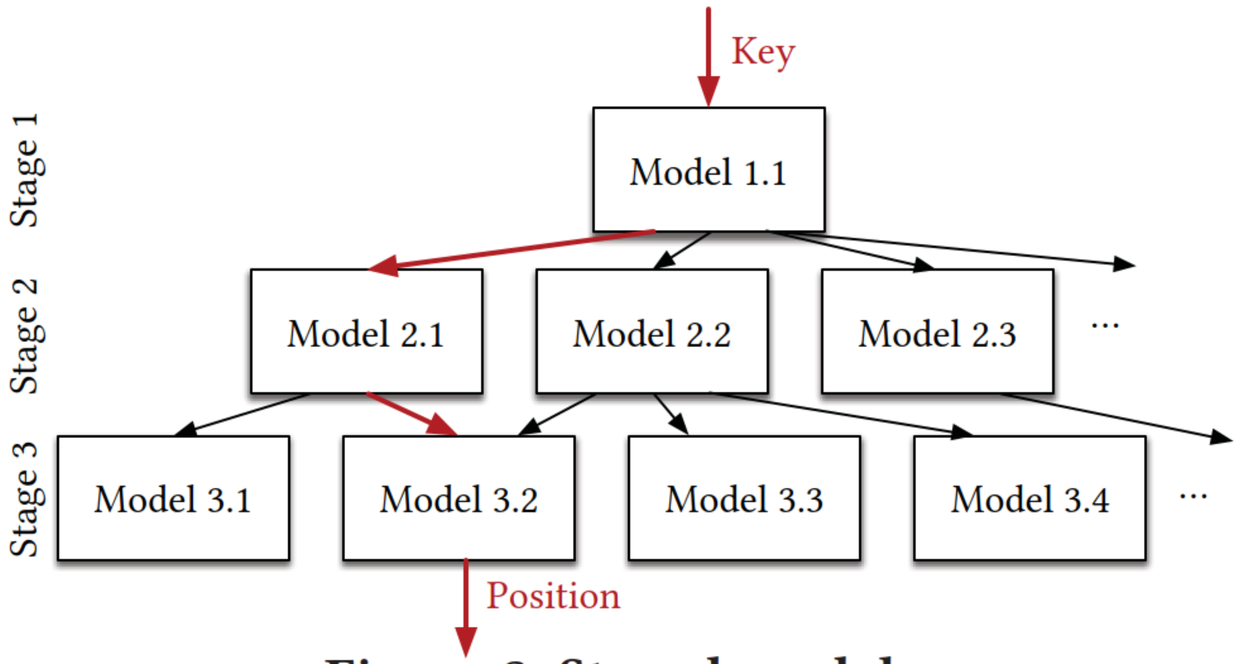


Figure 3: Staged models

The Recursive Model Index

- Benefits
  - Decouple execution cost & model size
  - Notion of progressively learning the shape of CDF
  - Divide the space into smaller ranges, easier to refine the final prediction

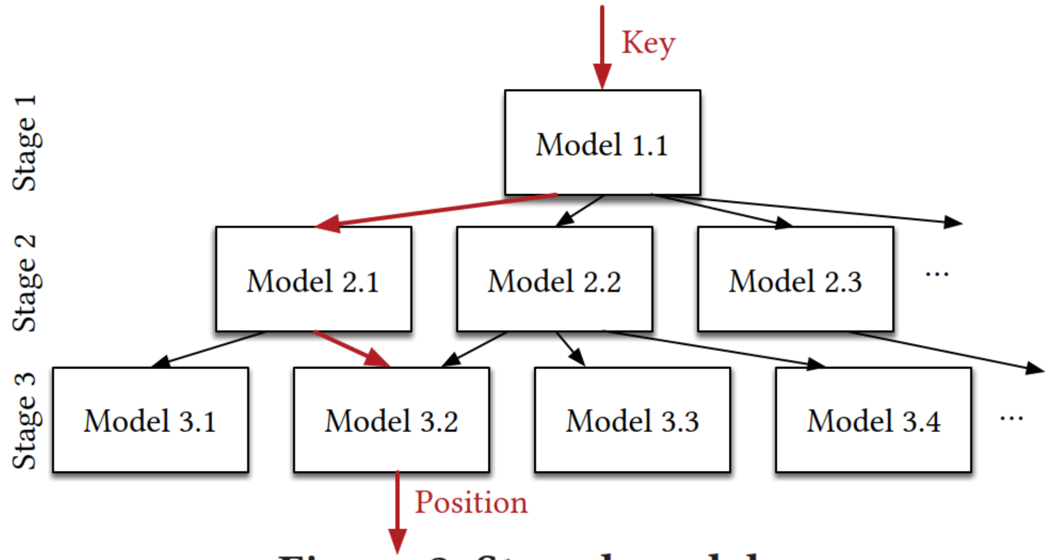


Figure 3: Staged models

The Recursive Model Index

- Worst case performance
  - If last stage models do not meet error requirement, replace by B-Trees
  - Have same worst case guarantee as B-Trees

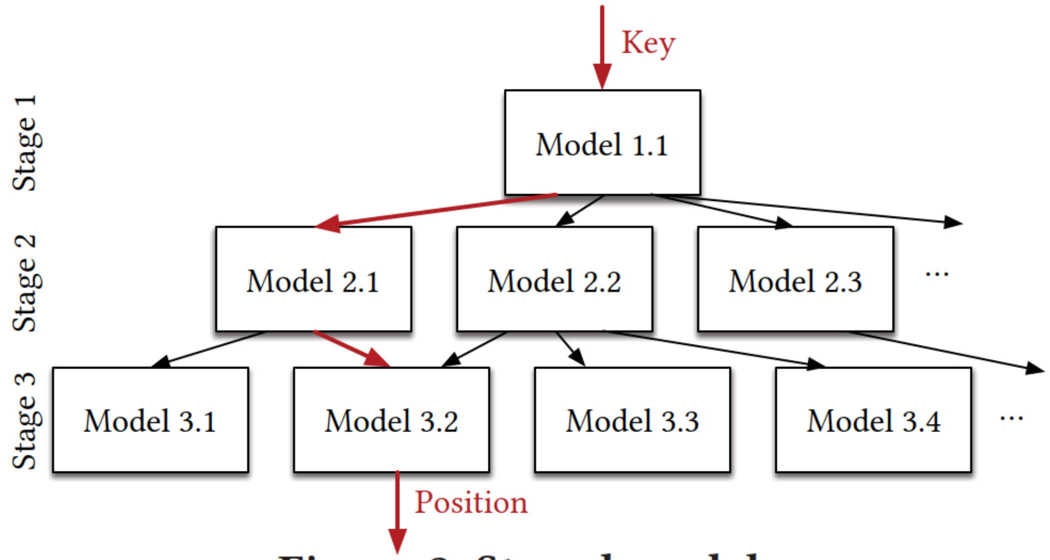


Figure 3: Staged models

The Recursive Model Index - Training

- Loss defined as:

$$L = \sum_{(x,y)} (f(x) - y)^2$$

- Simple model trained in seconds, Neural Nets in minutes

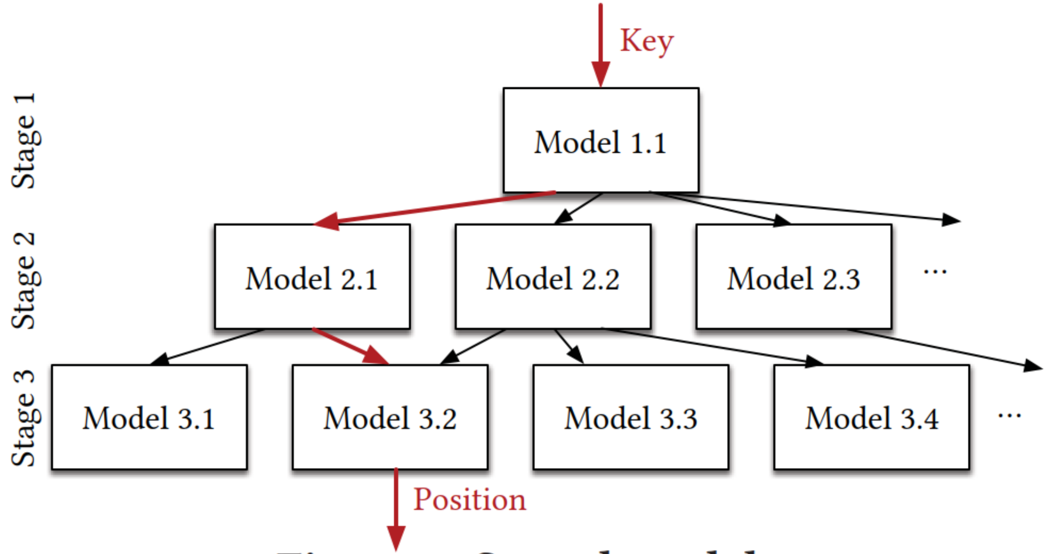


Figure 3: Staged models

Experiments

- Integer Datasets
  - Weblogs dataset contains 200M log entries
  - Maps dataset indexed the longitude of ≈ 200M user-maintained features
  - Log-normal dataset synthesized by sampling 190M unique values

- Models
  - 2-stage RMI model having second-stage sizes (10k, 50k, 100k, and 200k)
  - Read-optimized B-Tree with different page sizes

Results

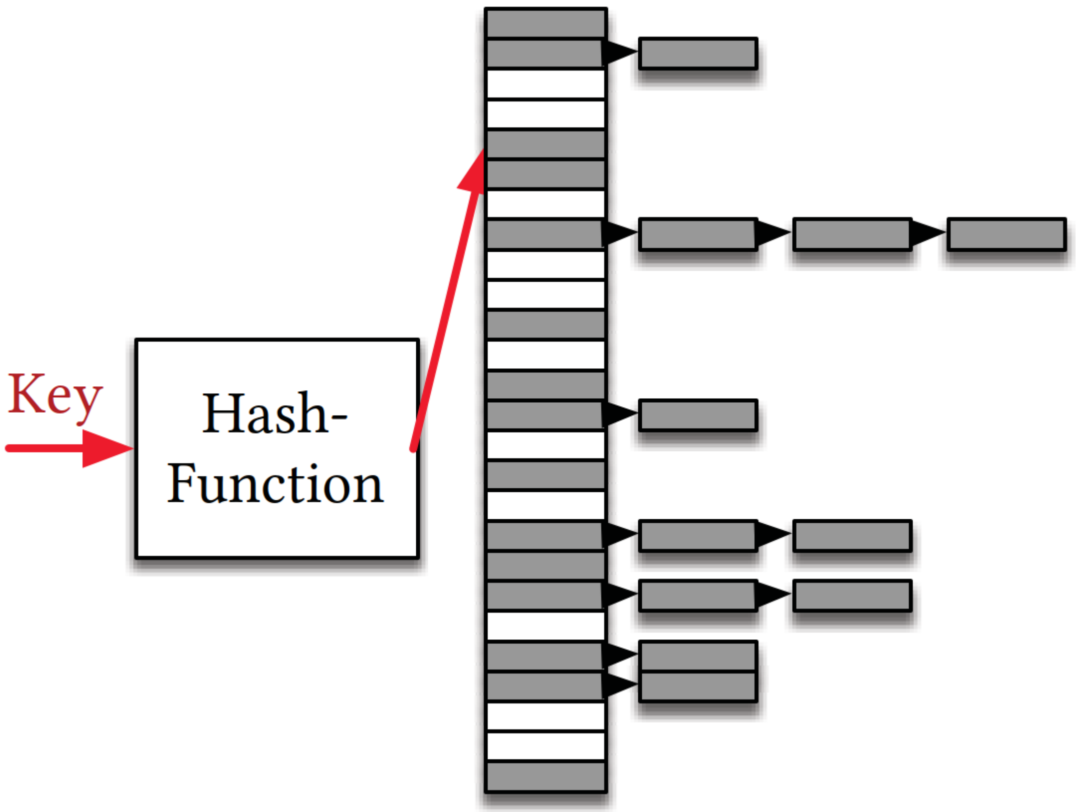
		Map Data			Web Data			Log-Normal Data		
Type	**Config**	Size (MB)	Lookup (ns)	Model (ns)	Size (MB)	Lookup (ns)	Model (ns)	Size (MB)	Lookup (ns)	Model (ns)
Btree	page size: 32	52.45 (4.00x)	274 (0.97x)	198 (72.3%)	51.93 (4.00x)	276 (0.94x)	201 (72.7%)	49.83 (4.00x)	274 (0.96x)	198 (72.1%)
	page size: 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%)

Figure 4: Learned Index vs B-Tree

Point Index

Point Index

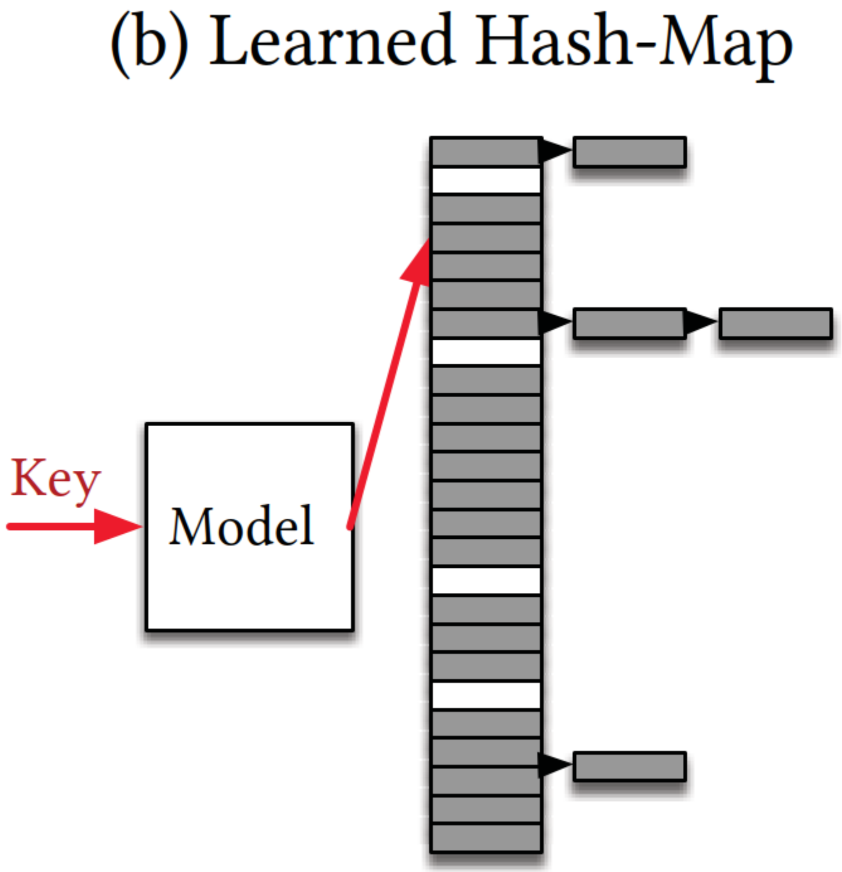
- Example: hash-map
- Deterministically map keys to positions inside an array



The Hash-Model Index

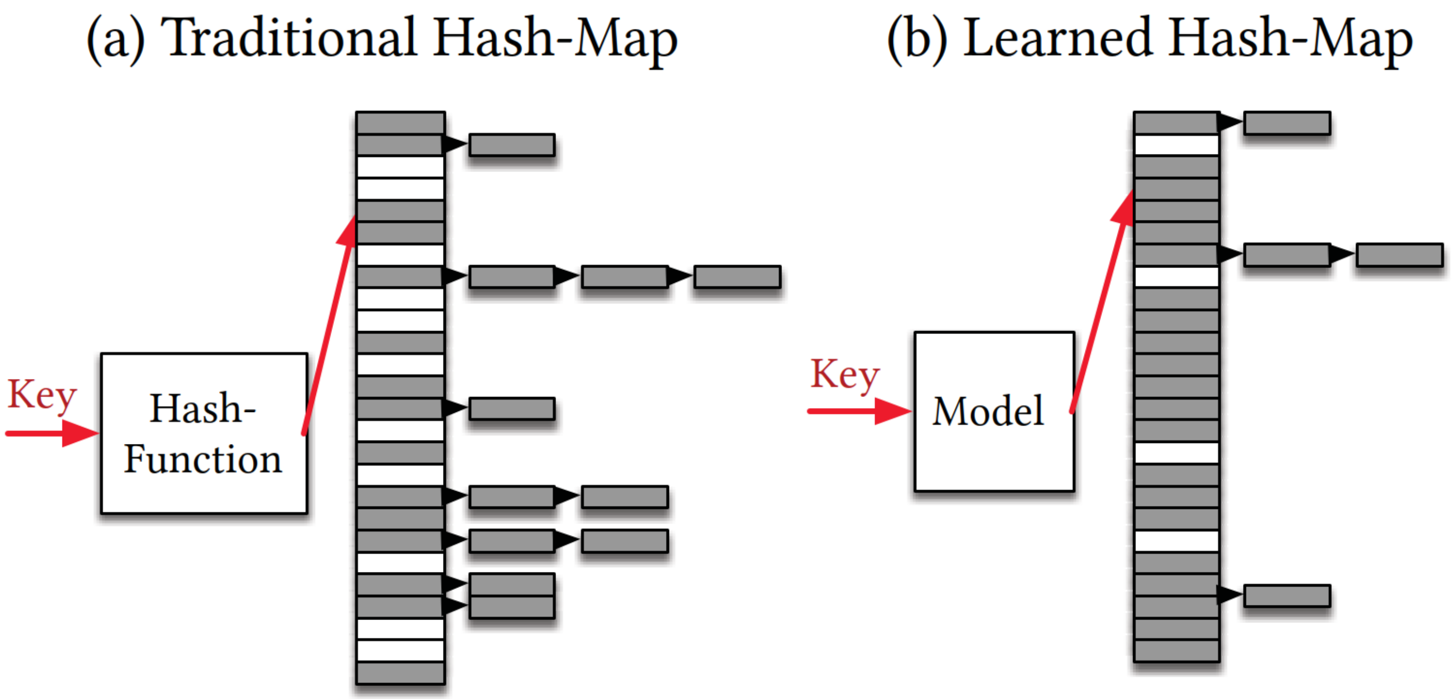
- Build a hash function based on the CDF of the data (M is size of hash-map):

$$h(key) = F(key) * M$$
$$F(key) \approx P(X \leq key)$$



The Hash-Model Index

- Main objective is to reduce number of conflicts
  - Conflicts could induce high cost depending on architecture (e.g. distributed)



Experiments

- Learned models with same settings as in range index
- Compared against MurmurHash3-like hash-function

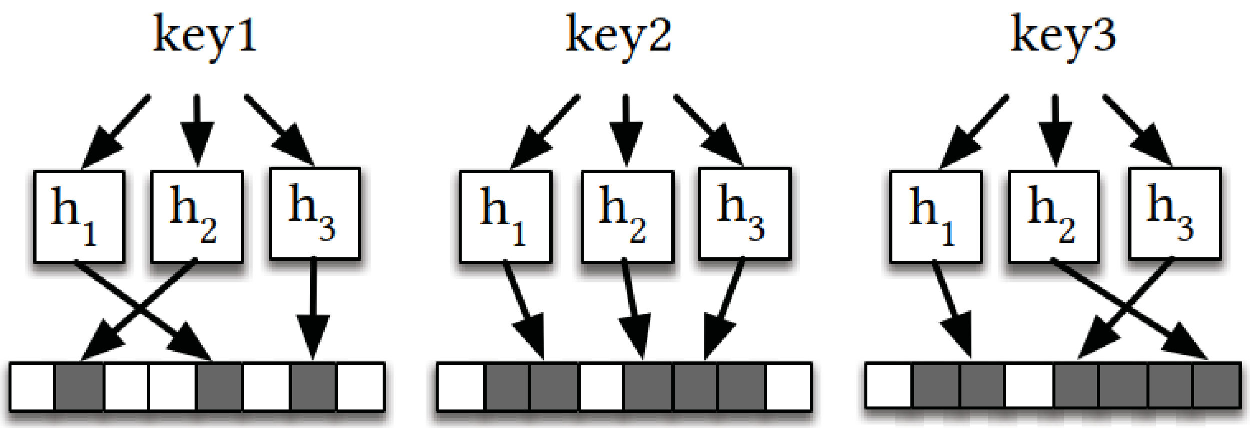
	% Conflicts Hash Map	% Conflicts Model	Reduction
Map Data	35.3%	07.9%	77.5%
Web Data	35.3%	24.7%	30.0%
Log Normal	35.4%	25.9%	26.7%

Figure 8: Reduction of Conflicts

Existence Index

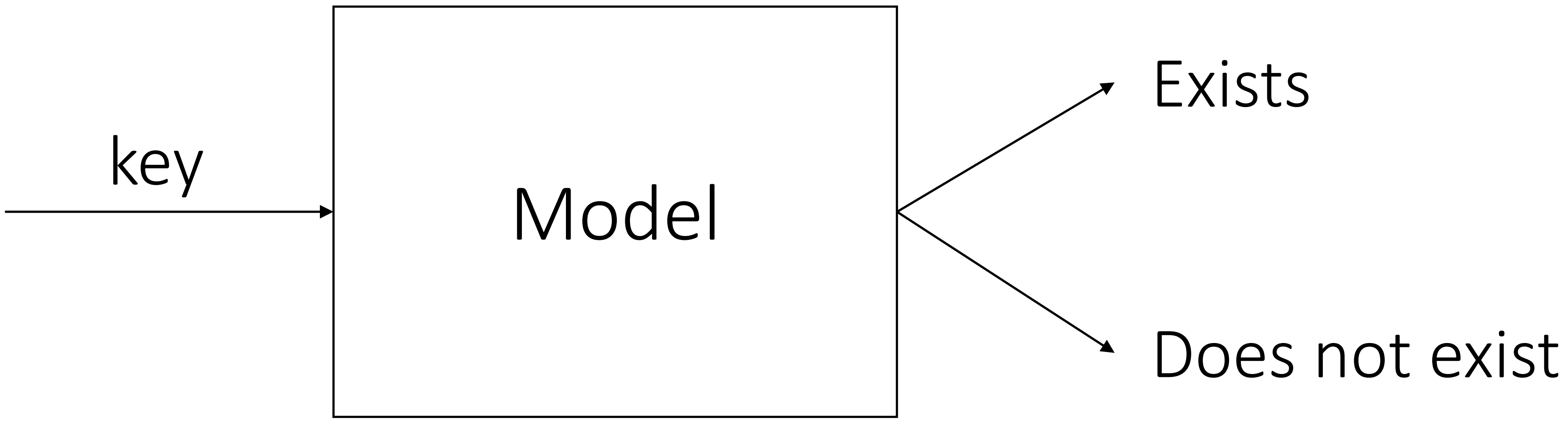
Existence Index

- Example: Bloom filters
- Return whether a key exists in a dataset
- No false negatives, but has potential false positives



Bloom filters as a Classification Problem

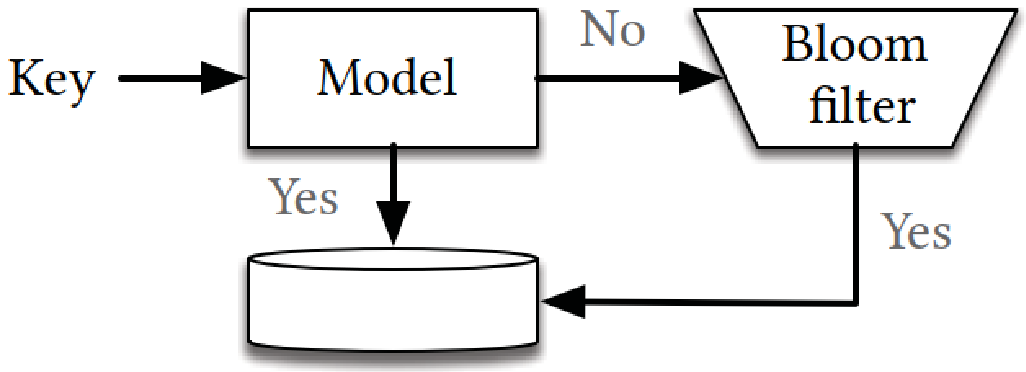
- Binary probabilistic classification task: Whether key exists in dataset



Bloom filters as a Classification Problem

- Guarantee for no false negative
- Overflow bloom filter: remember false negatives from models

(c) Bloom filters as a classification problem



Experiments

- Data
  - 1.7M blacklisted phishing URLs
  - Negative set: random URLs + whitelisted URLs
- Comparison
  - Learned filter: RNN with GRU
  - Normal Bloom filter

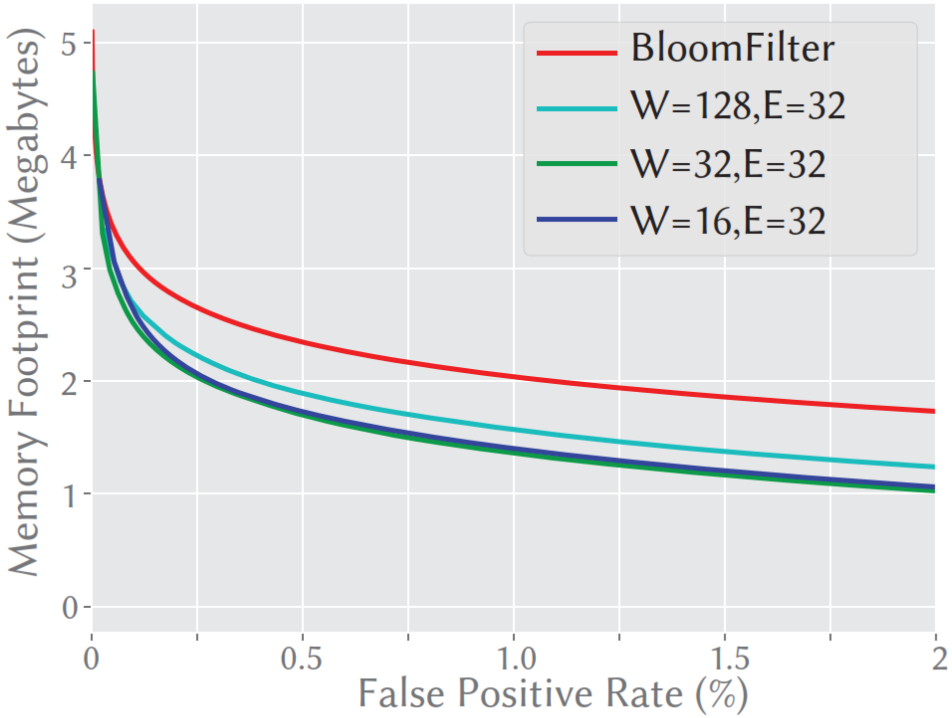


Figure 10: Learned Bloom filter improves memory footprint at a wide range of FPRs. (Here W is the RNN width and E is the embedding size for each character.)

Critique

Major Contributions

1. Proposed the idea of applying machine learning in index structures
2. Solutions to offering guarantees on performance, determinism with ML models
3. Showed significant performance improvements (time and space)
4. Inspired new research direction (27 citations since June 2018)

Criticism

1. Detail of platform used for experiments not given
2. Little discussion on training time
3. Experiments on CPU only

Conclusion & Future Direction

- Proposed a new direction in database research that
  - Makes effective use of machine learning methods
  - Shows promising preliminary results
  - Inspired new research work

- Requires more details on performance evaluation

- Potentials in learned algorithms, multi-dimensional indexes