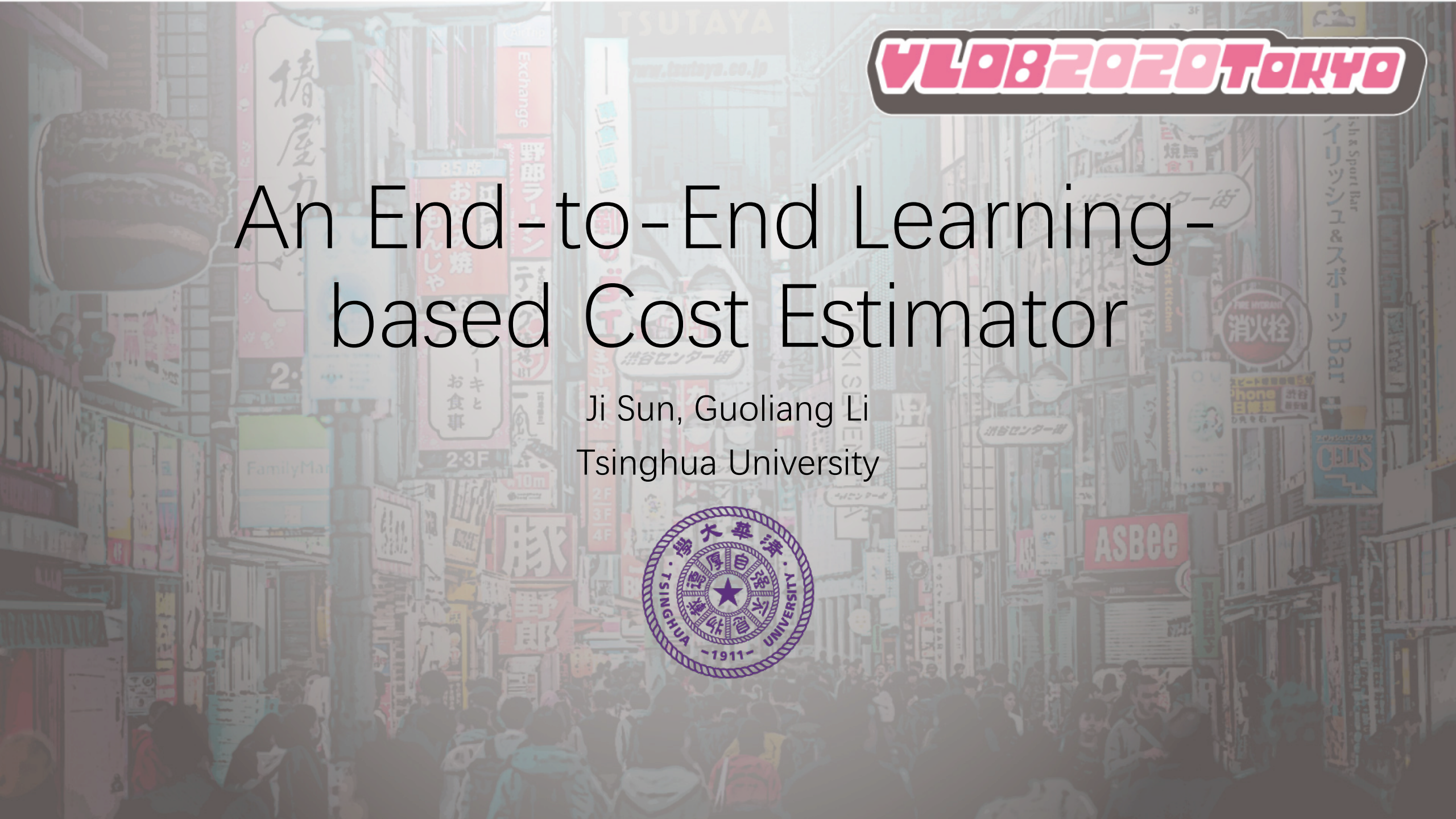
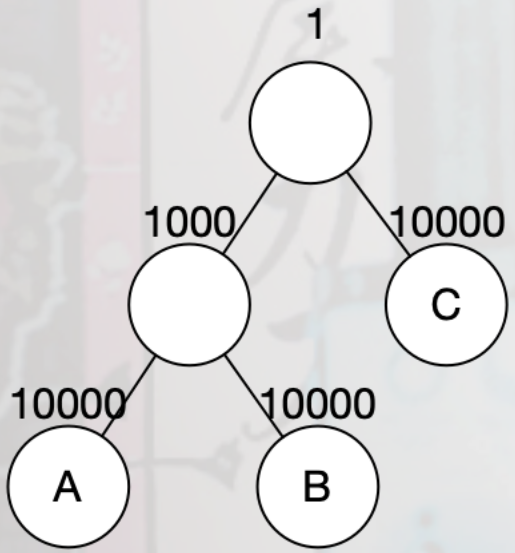


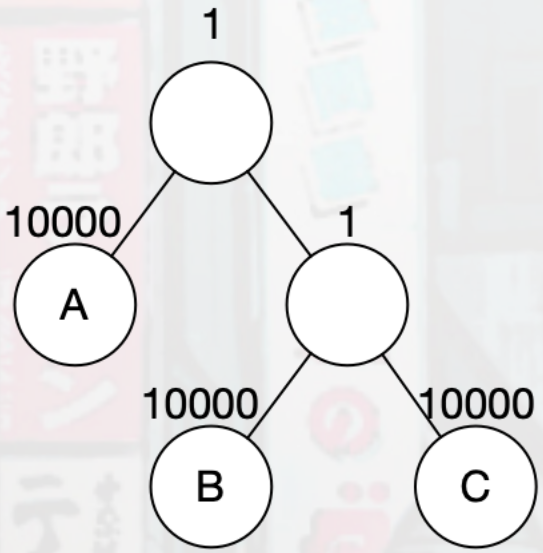
An End-to-End Learning-based Cost Estimator

Ji Sun, Guoliang Li

Tsinghua University

Plan Cost is Vital to Plan Selection





Cost Estimates Methods

- Cardinality Estimates → Cost Estimates
- Cardinality Estimates
  - Histogram + Assumption of Independence
  - Sampling
    - Random sampling
    - Index-based sampling
    - Weighted Sampling
  - Kernel Density Estimator
  - Autoregressive Model
  - Light-weighted neural network
- Cost Estimates
  - Handcrafted Function with Given Cardinality
  - Learned Function with Given Cardinality

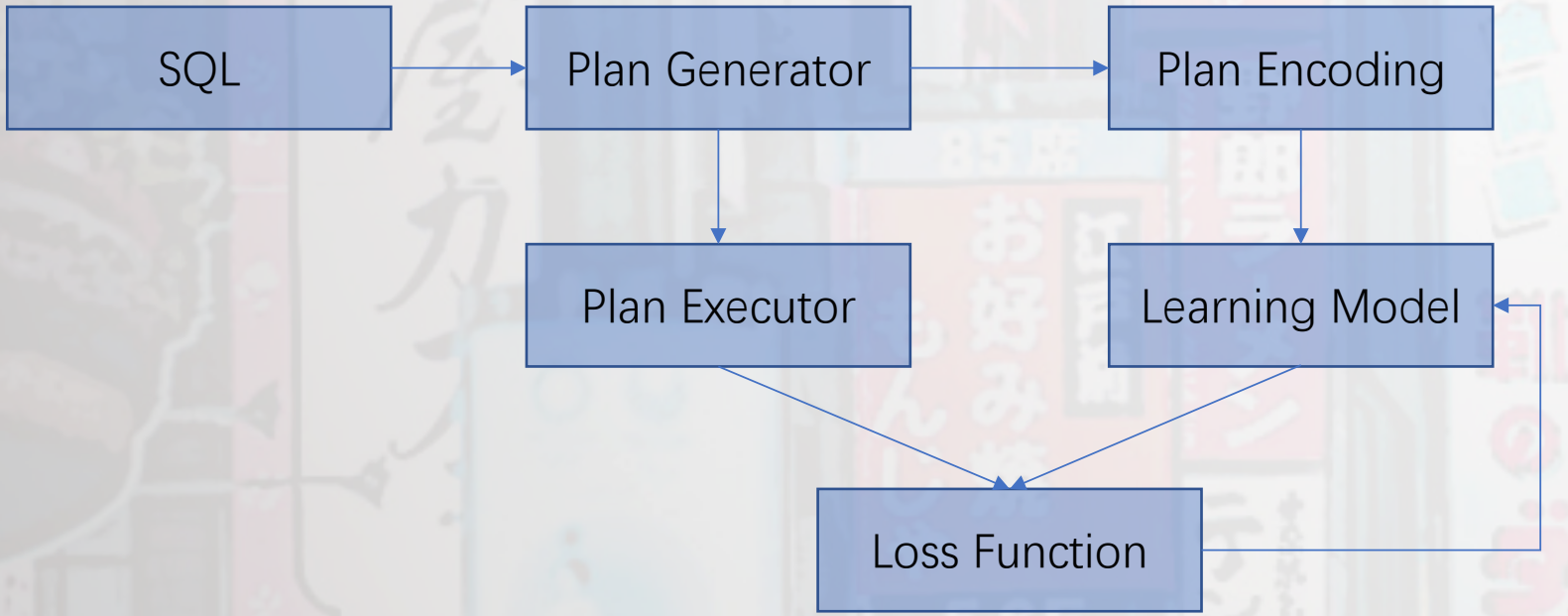
Depend on Data/Index

Depend on Queries

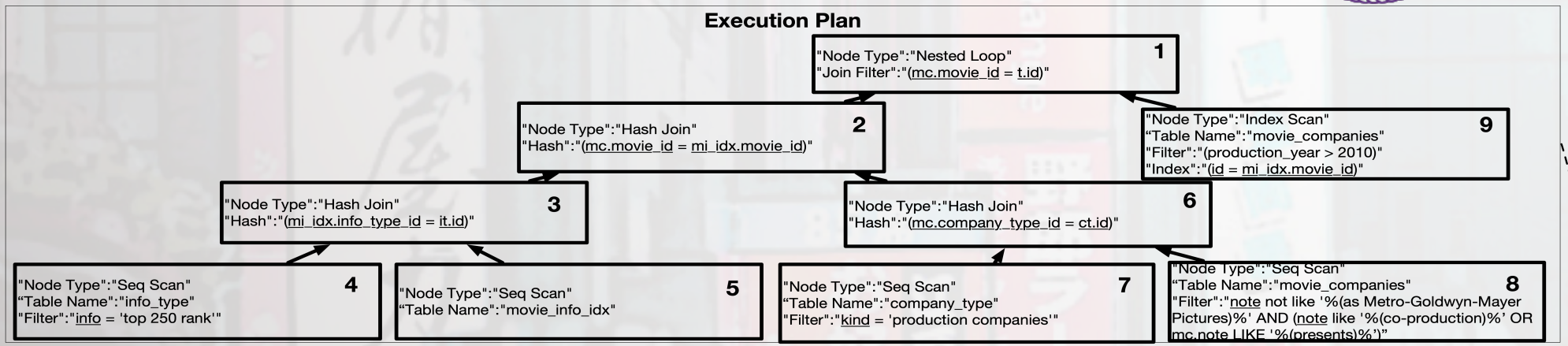
Why End-to-End Cost Estimator

Efficient	**Accurate**
Estimate the Plan Directly	Cardinality Helps Cost Learning
Cache Representation for Subplans	Learn Relations Between Queries and Latent Block Readings
Batch Training and Evaluation	Tree-structured Model Supports Deep Plans
	Substring Embedding Supports `LIKE` Predicates

Training Workflow



What We Need for Cost Estimates



One-hot Encoding

Operator	one-hot
=	0001
>	0010
not like	0100
like	1000

Operation	one-hot
Nested Loop	0001
Hash Join	0010
Seq Scan	0100
Index Scan	1000

Table Name	one-hot
info_type	0001
movie_info_idx	0010
company_type	0100
movie_companies	1000

Column	one-hot
mc.movie_id	000000000001
t.id	000000000010
mi_idx.movie_id	000000000100
mi_idx.info_type_id	000000001000
it.id	000000010000
it.info	000000100000
mc.company_type_id	000001000000
ct.id	000010000000
ct.kind	000100000000
mc.note	001000000000
mc.production_year	010000000000
mc.id	100000000000

Sample Datasets

info_type.info	movie_companies.note	movie_companies.production_year	company_type.kind
top 260 rank	(as Metro-Goldwyn-Mayer Pictures) (co-production)	1987	production companies
top 270 rank	(2006)(USA)(TV)	2013	distributors
top 250 rank	(2006)(worldwide)(TV)	1995	special effects companies
top 250 rank	(2011)(UK)(TV)	1966	miscellaneous companies

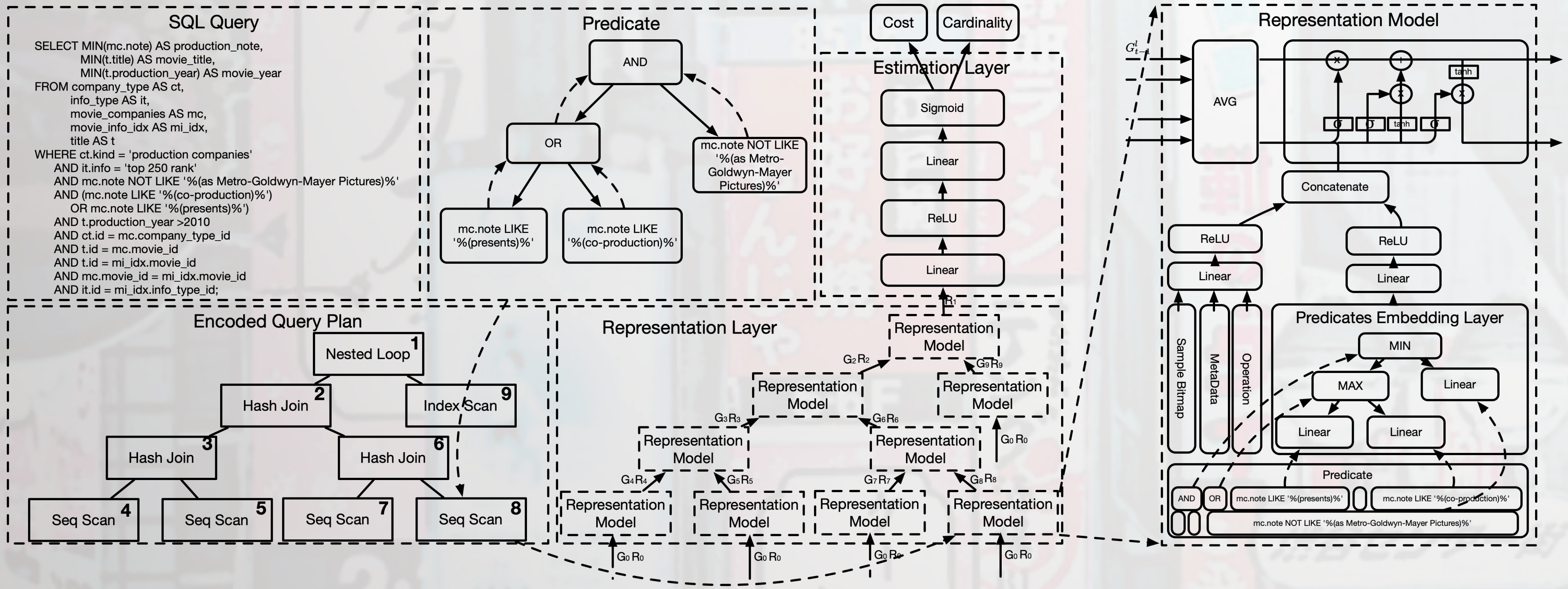
Dictionary

Token	Representation
'top 250 rank'	0.14,0.43,…,0.92
'production companies'	0.51,0.22,…,0.11
'(as Metro-Goldwyn-Mayer Pictures)'	0.91,0.35,…,0.25
'(co-production)'	0.37,0.11,…,0.02
'(presents)'	0.13,0.41,…,0.76

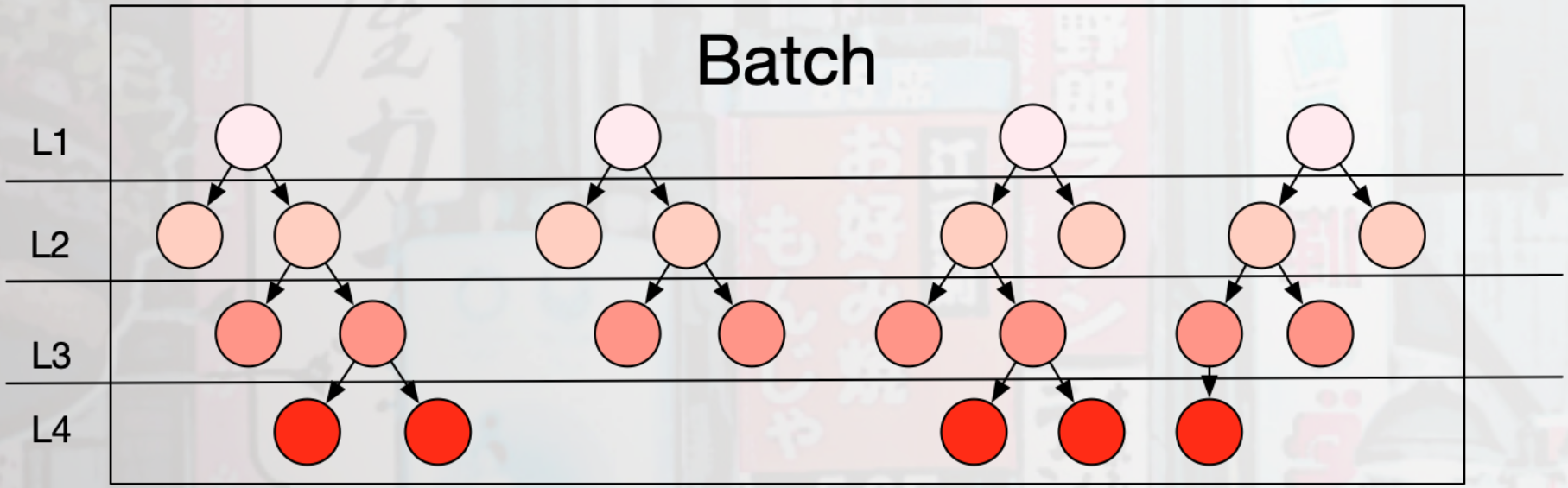
id	Operation	MetaData	Predicate	Sample Bitmap
1	0001	padding	000000000001 0001 000000000010	padding
2	0010	padding	000000000001 0001 000000000100	padding
3	0010	padding	000000001000 0001 000000010000	padding
4	0100	0001	000000100000 0001 0.14,0.43,…,0.92	0011
5	0100	0010	padding	1111
6	0010	padding	000001000000 0001 000010000000	padding
7	0100	0100	000100000000 0001 0.51,0.22,…,0.11	1000
8	0100	1000	001000000000 0100 0.91,0.35,…,0.25 001000000000 1000 0.37,0.11,…,0.02 001000000000 1000 0.13,0.41,…,0.76	1000
9	1000	1000	010000000000 0010 0.81 100000000000 0001 00000000100	0100

How the Model Works

- Local Information is embedded by NNs.
- Global Information is composed of local information and previous global information.
- Cardinality & Cost errors are considered when conducting backwards training.



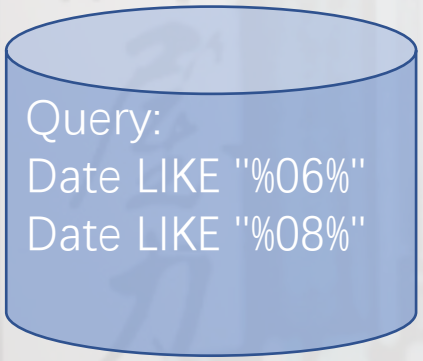
How to Accelerate the Training



How to Encode Substring in `LIKE`

- Motivation:
  - Sparse Space: One-hot encoding is too sparse and pointless
  - High Overhead: Number of substrings in database is huge
- Approach:
  - Extracting substrings from existing queries
  - Finding general rules for extracting these substrings from data
  - Extracting all possible substrings from data by rules set.
  - Embedding all substrings by using Skip-gram.

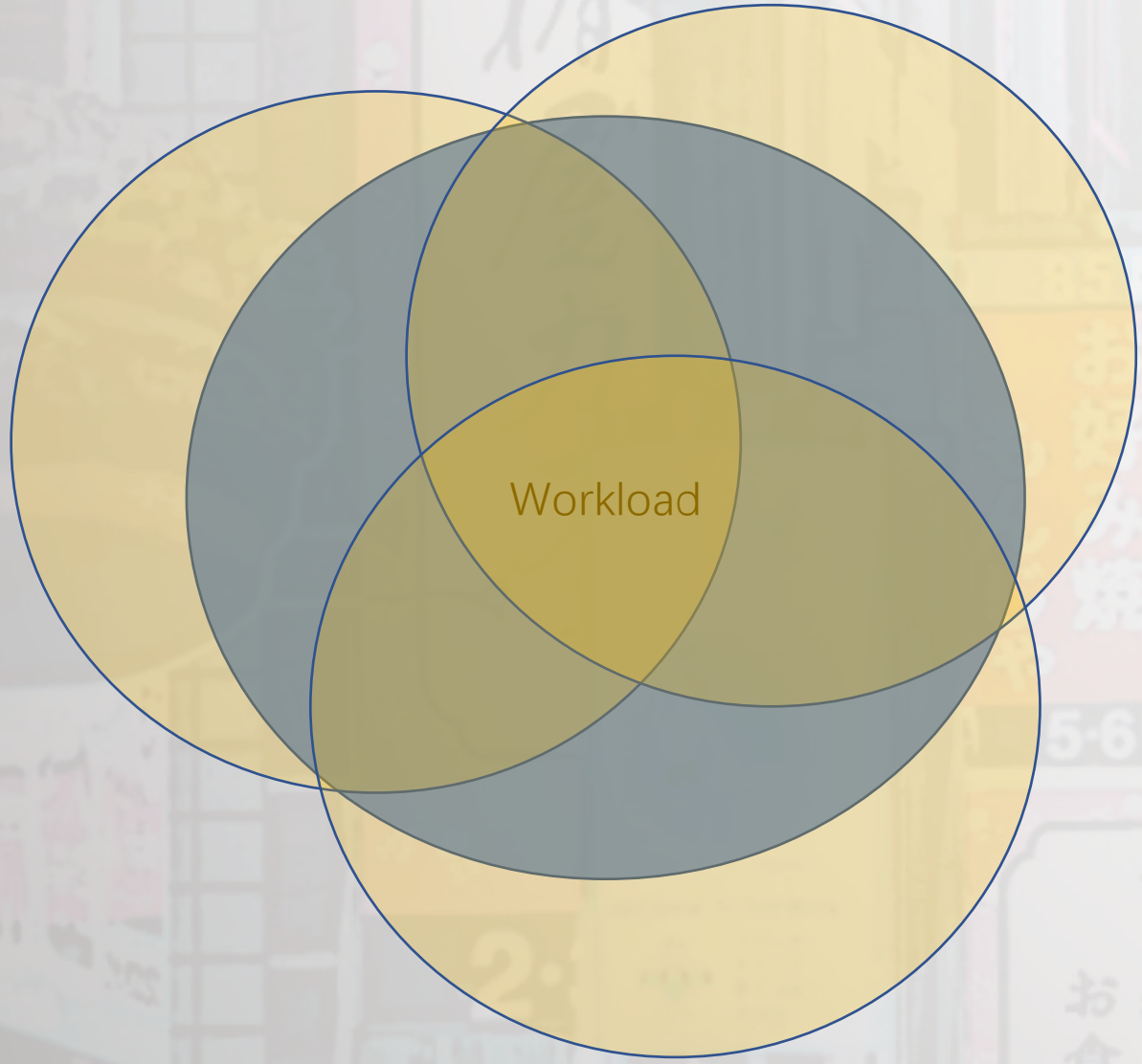
How to Select the Best Rule(s)



rule = ⟨F,P,L⟩,
F: {Prefix, Suffix}
P ∈ combination{PC , Pl, Ps, Pn, Pt(T)}
PC = [A–Z]+;
Pl = [a–z]+;
Ps = whitespace+;
Pn = [0–9]+;
Pt(T) = T;

Pattern	Function	Extracted Length	Extracted Strings	Utility/Redundancy
Pt(`06`)	Prefix	2	06	0.5/0.0
Pn	Prefix	2	20 06 29 08 26	1.0/0.4
Pt(`(`)PnPt(`-`)Pn	Suffix	2	06 08	1.0/0.0 ✓

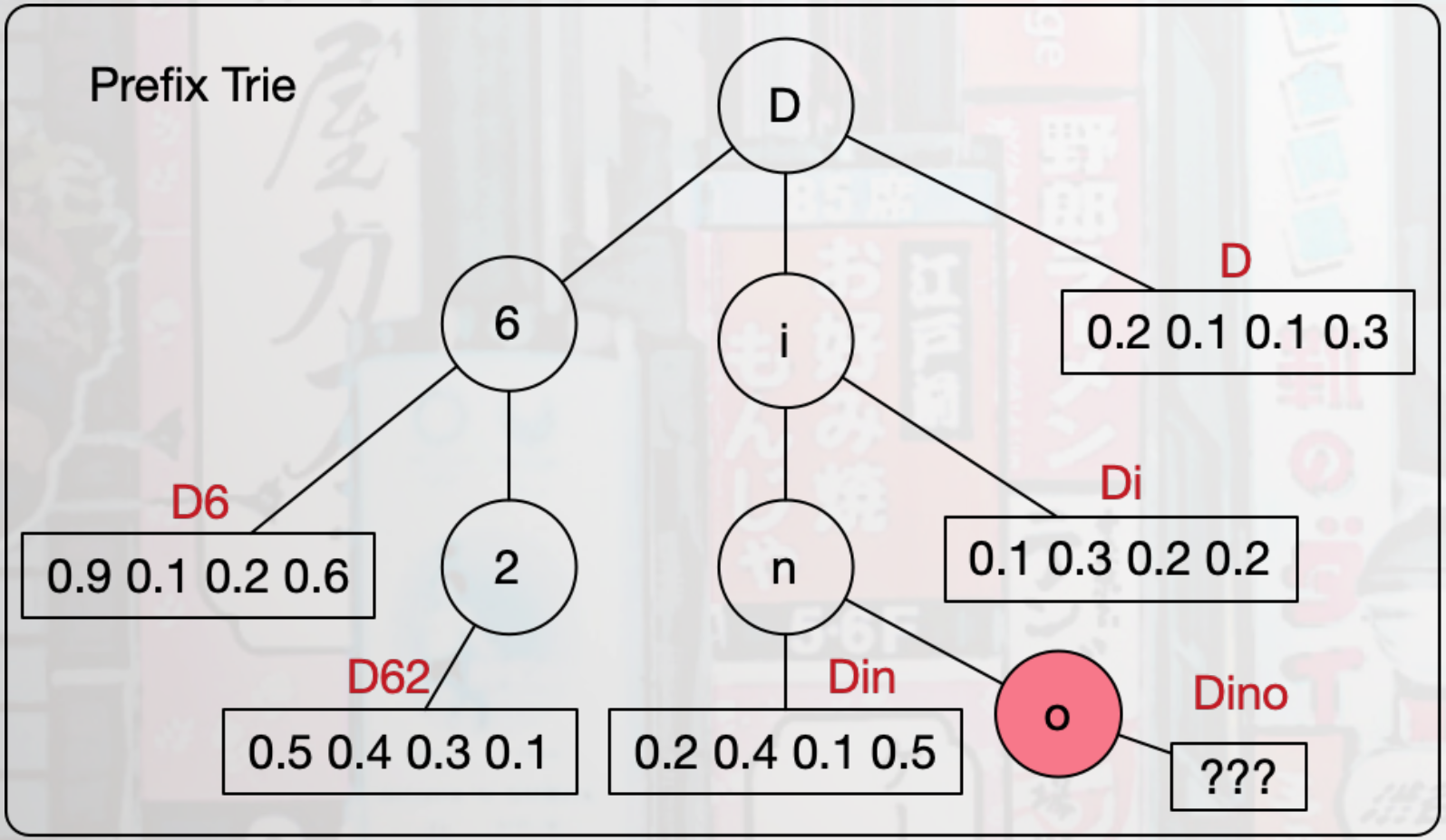
A Set Cover Problem



Substrings Extracted by a Rule

Substrings in LIKE Predicates

How to organize the Embeddings



Experiments

Dataset

We use the real dataset IMDB and JOB workload.

Workload	#Queries	#Joins	String Predicates
Synthetic	5000	0-2	NO
Scale	500	0-4	NO
JOB-light	700	1-4	NO
JOB	113	5-15	YES

Environment

We use a machine with Intel(R) Xeon(R) CPU E5-2630 v4, 128GB Memory.

Experiments

Methods

Methods	Representation Network	Predicate Network	Estimate Network	String Encoding	Sample Bitmap
PostgreSQL	NO	NO	NO	NO	NO
Mysql	NO	NO	NO	NO	NO
Oracle	NO	NO	NO	NO	NO
MSCN	NO	CNN	Card	NO	YES/NO
TNN	FC	LSTM	Card/Cost	Rule+Emb	YES/NO
TLSTM	LSTM	LSTM	Card/Cost	Rule+Emb	YES/NO
TPool	LSTM	MinMaxPool	Card/Cost	Rule+Emb	YES/NO

Experiments

JOB-light Workload

Cardinality	median	90th	95th	99th	Max	mean
PostgreSQL	7.93	164	1104	2912	3477	174
MySQL	9.55	303	685	2256	2578	149
Oracle	8.32	374	976	2761	3331	157
MSCN	3.82	78.4	362	927	1110	57.9
TNN	2.95	76.8	275	799	902	49.8
TLSTM	3.73	50.8	157	256	289	24.9
TPool	3.51	48.6	139	244	272	24.3

Cost	median	90th	95th	99th	Max	mean
PostgreSQL	26.8	332	696	2740	3020	173
MySQL	9.47	102	342	1293	2228	84.5
Oracle	12.3	157	278	1366	1825	102
MSCN	4.75	11.3	40.1	563	987	27.4
TNN	2.06	25.5	134	293	401	19.1
TLSTM	3.66	32.1	80.3	445	583	17
TPool	1.85	11.1	20.3	101	125	5.76

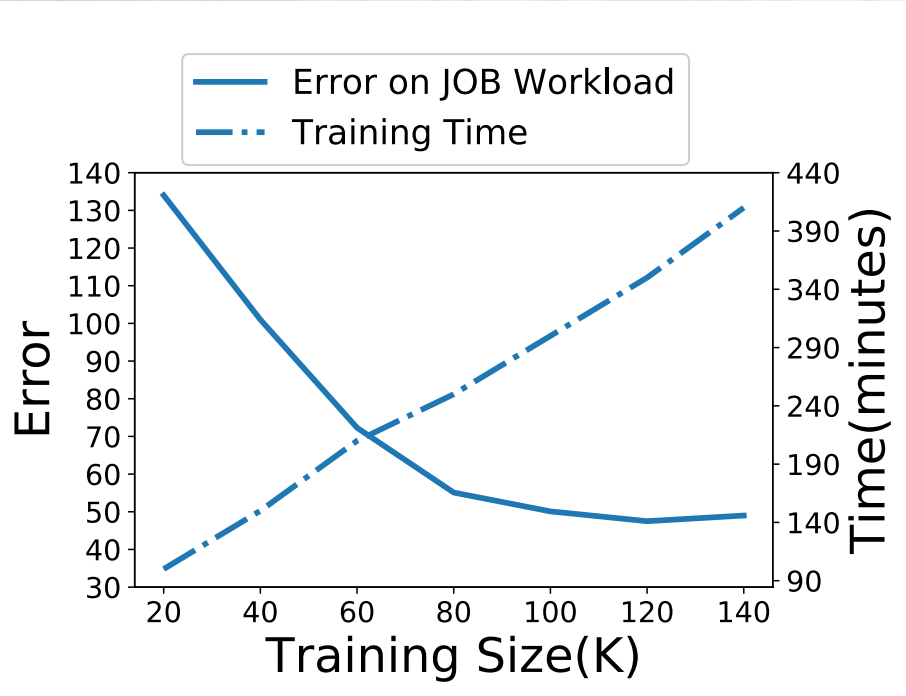
Experiments

JOB Workload

Cardinality	median	90th	95th	99th	Max	mean
PostgreSQL	184	8303	34204	106000	670000	10416
TLSTM-Hash	11.1	207	359	824	1371	83.3
TLSTM-Emb	11.6	181	339	777	1142	70.2
TLSTM-EmbRule	10.9	136	227	682	904	55.0
TPool	10.1	74.7	193	679	798	47.5

Cost	median	90th	95th	99th	Max	mean
PostgreSQL	4.90	80.8	104	3577	4920	105
TLSTM-Hash	4.47	53.6	149	239	478	24.1
TLSTM-Emb	4.12	18.1	44.1	105	166	10.3
TLSTM-EmbRule	4.28	13.3	22.5	104	126	8.6
TPool	4.07	11.6	17.5	63.1	67.3	7.06

Experiments



JOB Training Errors

Method	Time (ms)
PostgreSQL	18.9
MSCN	14.2
TLSTM	70.3
TLSTMBatch	3.63

JOB Average Evaluation Time

Ji Sun
sun-j16@mails.tsinghua.edu.cn

- **An End-to-End Learning-based Cost Estimator**
  - **Effective Feature Extraction & Encoding**
  - **Recursive Tree Model & Batch Training**
  - **Rule-based Substring Extraction & String Embedding**

- **Special Thanks To:**