深圳北理莫斯科大学
УНИВЕРСИТЕТ МГУ–ППИ В ШЭНЬЧЖЭНЕ
SHENZHEN MSU–BIT UNIVERSITY

Taste: Towards Practical Deep Learning-based Approaches for Semantic Type Detection in the Cloud

Dr. Feng Liang
Shenzhen MSU-BIT University
fliang@smbu.edu.cn

Content

- Background
- Problem
- Method
- Evaluation
- Discussion

Background: Tabular Semantic Type Detection

Tabular semantic types uncover the semantic meaning of table data and usually map to real-world concepts or entities.

Example semantic types: credit card number, phone number, address, city, email, numbers related to ID (social security, passport, ...), job, etc

Applications:

- **Table understanding**
- **Data cataloging and searching**
- **Data quality validation**
- **Data transformation**
- **Data wrangling**
- **...**

Background: Tabular Semantic Type Detection

深圳北理莫斯科大学
УНИВЕРСИТЕТ МГУ–ППИ В ШЭНЬЧЖЭНЕ
SHENZHEN MSU–BIT UNIVERSITY

Commercial products:

- Alteryx Trifacta
- Microsoft PowerBI
- Tableau

Cloud service providers:

- Microsoft Purview
- AWS Glue Data Catalog
- Google Looker Studio
- Alibaba Cloud Data Security Center

alteryx TRIFACTA

tableau

Microsoft

aws

Google Cloud

Alibaba Cloud

Problem

Rely on table content -> Intrusive to user data sources:

- **Increased I/O and connections on user data**
- **Concerns about data leakage, auditing, and compliance**

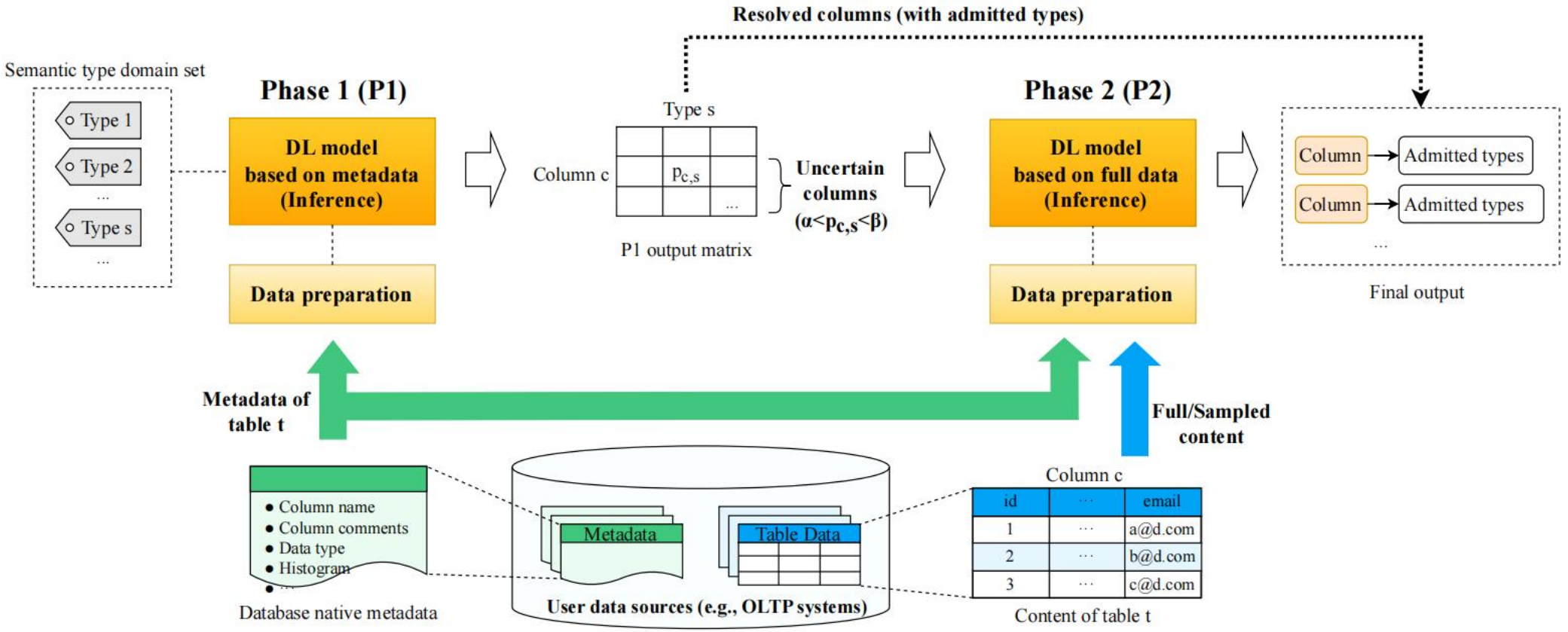
Execution-inefficient:

- **Column scanning operations are costly (as high as billions of tables and columns from diverse tenants)**
- **Abundant column features increase the complexity of the DL model.**

Our solution: TASTE

深圳北理莫斯科大学
УНИВЕРСИТЕТ МГУ–ППИ В ШЭНЬЧЖЭНЕ
SHENZHEN MSU–BIT UNIVERSITY

TASTE: two phase semantic type detection with minimal access to table content.



Our solution: TASTE

Features:

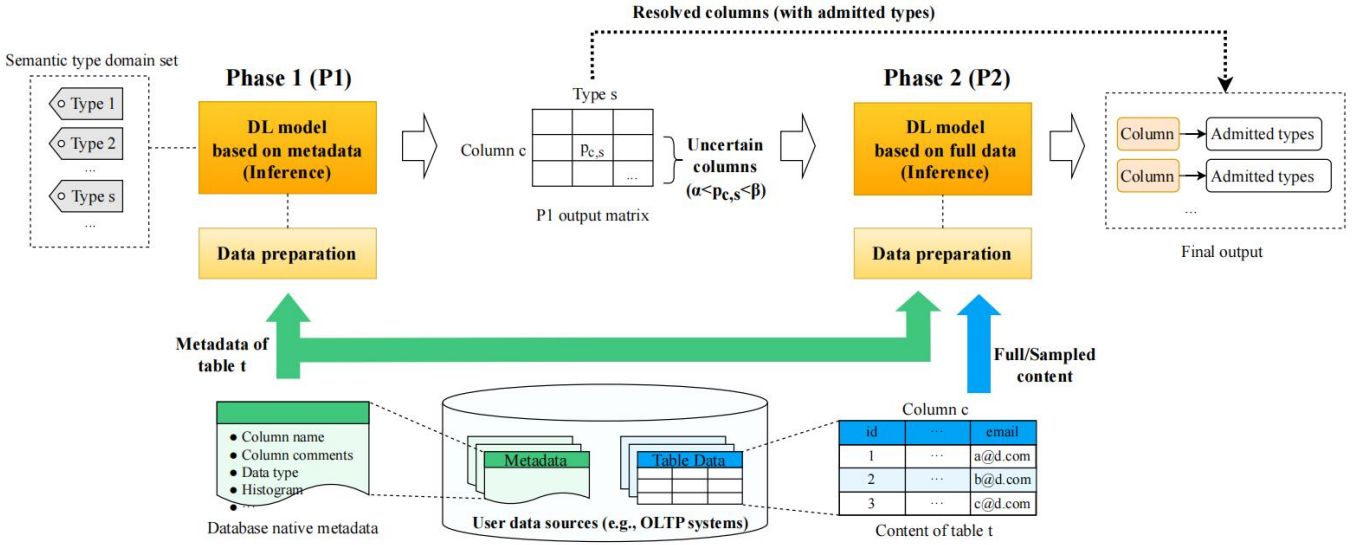
- **Two detection phases with different computation complexity and prediction precision (alleviating intrusiveness and increasing compuational efficiency)**
- **A novel multi-task learning model to accelerate two phase design (increasing computational efficiency)**
- **An efficient pipeline implementation (ibid)**

TASTE Feature: Two Phases

深圳北理莫斯科大学
УНИВЕРСИТЕТ МГУ–ППИ В ШЭНЬЧЖЭНЕ
SHENZHEN MSU–BIT UNIVERSITY

Phase 1 (P1), for each column:

- **Only input table metadata (via API or SQL on schema)**
- **Conclude the semantic type if confident**
- **Go through Phase 2 if not certain**



Phase 2 (P2), for each uncertain column:

- **Input table metadata and content (via full scan or sampling);**

TASTE Feature: Two Phases

深圳北理莫斯科大学
УНИВЕРСИТЕТ МГУ–ППИ В ШЭНЬЧЖЭНЕ
SHENZHEN MSU–BIT UNIVERSITY

Q: How to judge confident or uncertain?
A: For a semantic type s, a column c, and two predefined probability values, $\alpha < \beta$,
Uncertain: $\alpha < p_{c,s} < \beta$
Confident: $p_{c,s} \geq \beta$

Q: What are the final detected semantic types?
A: Confident types in P1+ types in P2

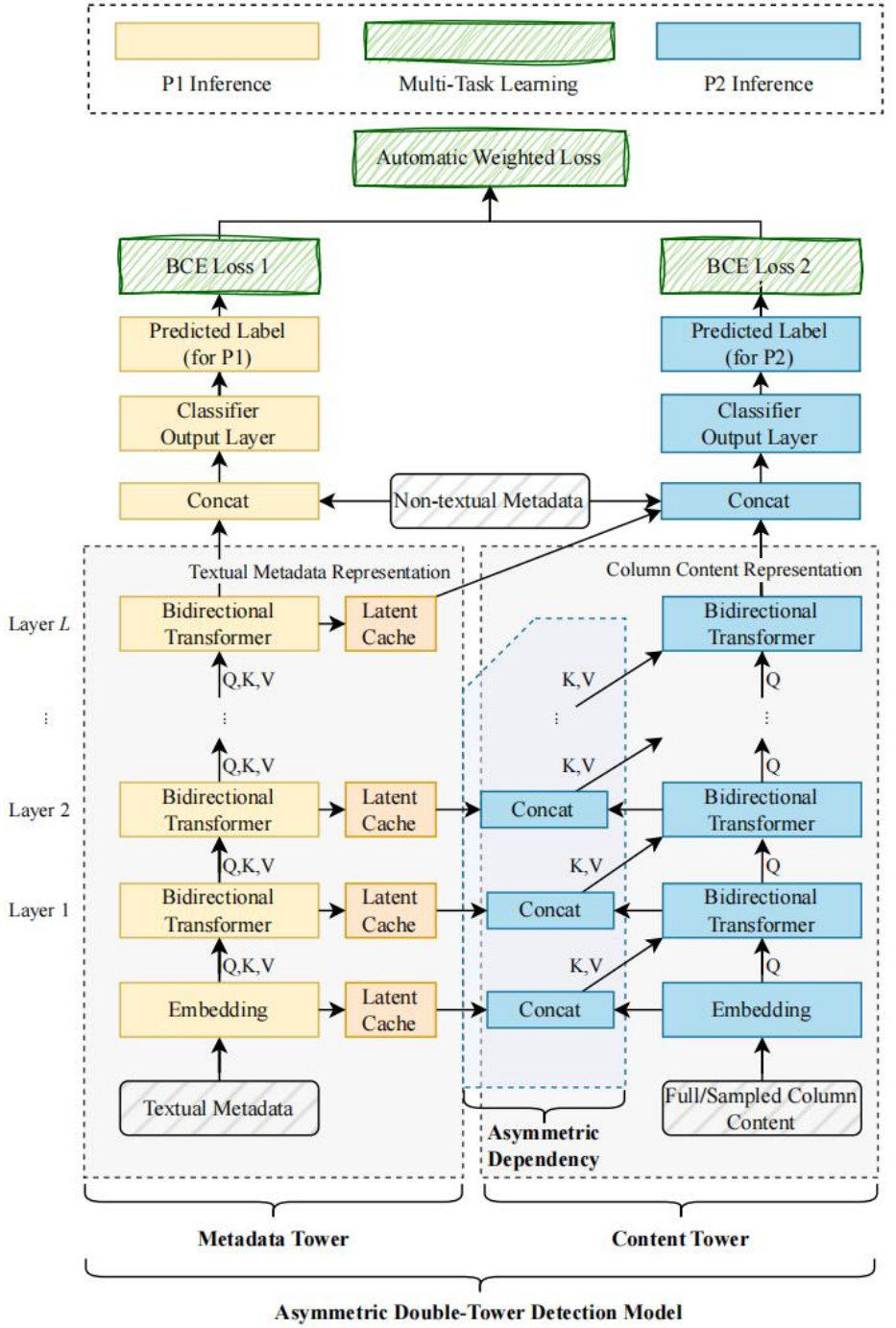
TASTE Feature: Two Phases

Q: How is intrusiveness alleviated?
A: For a column, if the metadata alone is enough for a confident semantic type result, there is no need to access table content in P2.

Q: How computational efficiency is achieved?
A: 1) There is a chance to skip P2, especially for columns that can easily be distinguished by the column name and comment.

TASTE Feature: the Asymmetric Double-Tower Detection (ADTD) Model

- **Two logical towers** with multiple layers of Transformers: the metadata and content towers.
- The content tower **asymmetrically relies on** the intermediate results of the metadata tower to address the attention of metadata and content.
- **Caching metadata intermediate** results for effeciency in P2
- These two towers use **shared parameters** and are **pretrained** with NLP tasks.



TASTE Feature: the ADTD Model

- **The metadata tower for P1:**

$$f_1(c) = Classify_{meta}(Encode_4^{\mathcal{M}_t^c} \oplus \mathcal{M}_n^c)$$

- **The metadata + content towers for P2:**

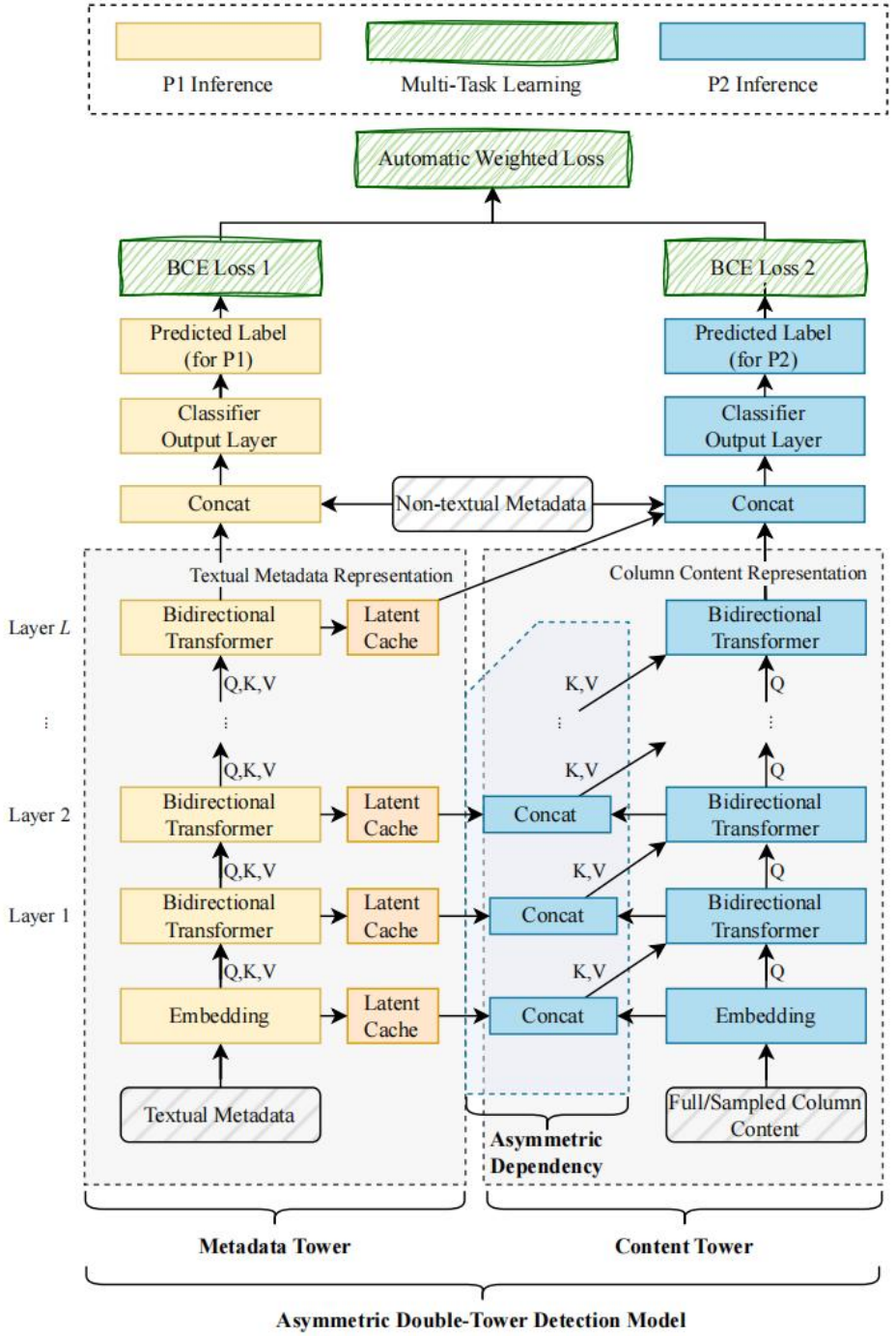
$$f_2(c) = Classify_{cont}(Encode_4^{\mathcal{D}^c} \oplus Encode_4^{\mathcal{M}_t^c} \oplus \mathcal{M}_n^c)$$

- **The single-task objective**

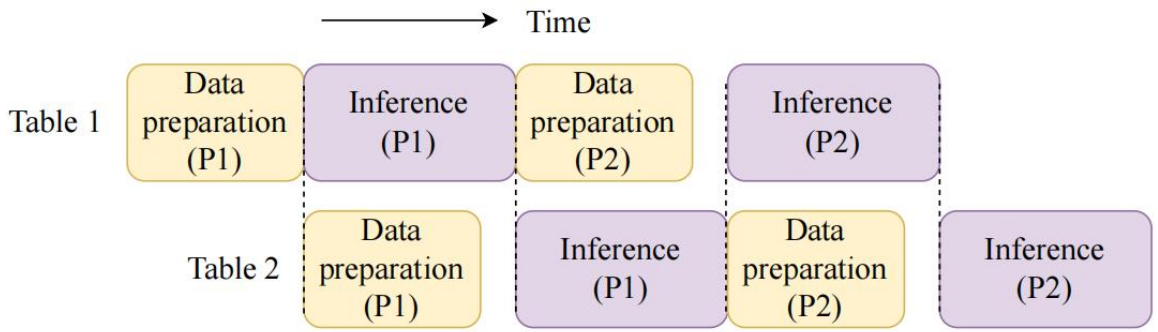
$$\mathcal{L}_{BCE}(p, y) = -\frac{1}{b}\sum_{c\in C}\sum_{s\in S} y_{c,s} log(p_{c,s}) + (1 - y_{c,s}) log(1 - p_{c,s})$$

- **The multi-task objective**

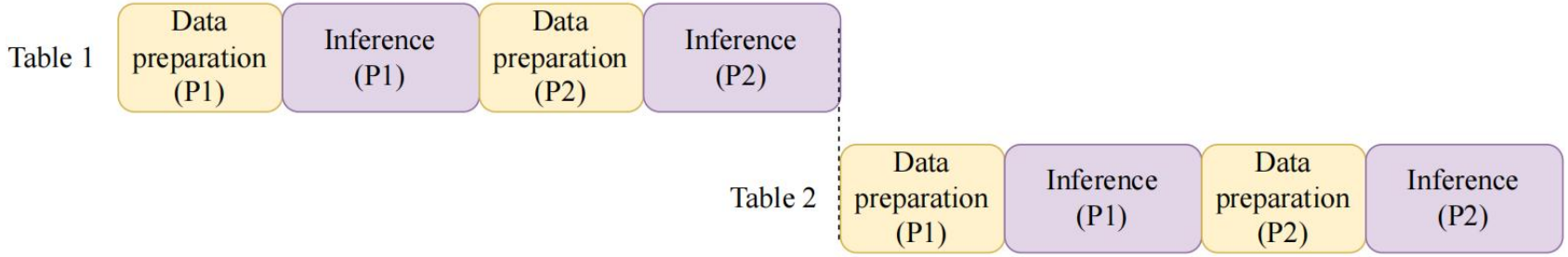
$$\mathcal{L}_{ADTD}(\mathcal{L}_{BCE}) = \sum_{i=1}^{2} \frac{1}{2w_i^2}\mathcal{L}_{BCE_i} + ln(1 + w_i^2)$$



TASTE Feature: the Pipeline Implementation



(a) Pipelined execution



(b) Sequential execution

Parallel stages during inference:

- **P1 ~ P2**
- **Data preparation ~ type inference**

Evaluation: Datasets

Table 2: Summary of the open datasets

Dataset	# tables	# cols	# types	% col w/o types
WikiTable	406,706	654,670	255	0%
- training	397,098	628,254	255	0%
- validation	4,764	13,025	248	0%
- testing	4,844	13,391	248	0%
GitTables-100K	100,000	1,212,987	1,953	31.56%
- training	80,000	966,107	1,884	31.43%
- validation	10,000	122,331	1,239	31.98%
- testing	10,000	124,549	1,289	32.13%

Evaluation: End-to-End Execution Time

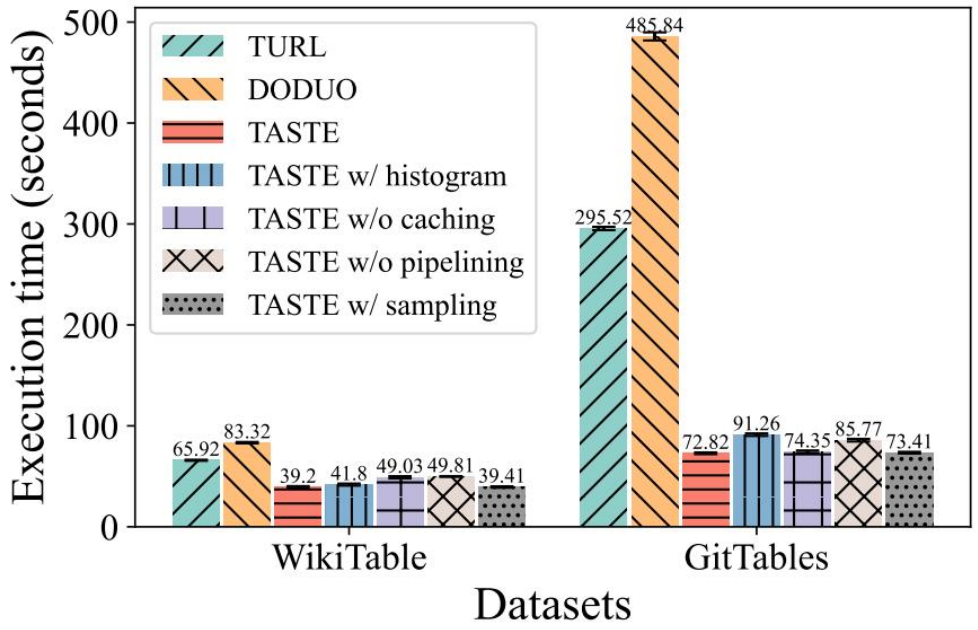


Figure 4: End-to-end execution time of different DL-based semantic type detection approaches.

Models are deployed on an ECS (pi7.8xlarge.4 type with 32 vCPU, 128 GB RAM and 2 NVIDIA A10 GPUs), and the test data are stored in an RDS for MySQL (version 8.0 of general-purpose family with 8 vCPU, 16 GB RAM and 500 GB SSD). Both the ECS
and RDS instances reside in one VPC (Virtual Private Cloud)

Using metadata for semantic type detection saves much time. The metadata intermediate caching and pipeline implementation optimize the computational efficiency.

Evaluation: F1 score

深圳北理莫斯科大学
УНИВЕРСИТЕТ МГУ–ППИ В ШЭНЬЧЖЭНЕ
SHENZHEN MSU–BIT UNIVERSITY

Table 3: F1 scores of different DL-based semantic type detection approaches on WikiTable and GitTables datasets ($n = 10$ and $l = 20$). For TASTE and its variants, $\alpha = 0.1$ and $\beta = 0.9$.

Model	Precision	Recall	F1
WikiTable dataset			
TURL	0.9275	0.9263	0.9269
DODUO	0.9325	0.9234	0.9279
TASTE	0.9344	0.9267	0.9306
TASTE w/ histogram	0.9414	0.9267	**0.9340**
TASTE w/ sampling	0.9342	0.9271	0.9306
GitTables dataset			
TURL	0.9852	0.9767	0.9809
DODUO	0.9923	0.9873	0.9898
TASTE	0.9947	0.9842	0.9894
TASTE w/ histogram	0.9957	0.9862	**0.9909**
TASTE w/ sampling	0.9945	0.9841	0.9893

TASTE achieves the SOTA prediction performance:

- **More abundant metadata**
- **Better cross-attention among columns**

Evaluation: F1 score with only metadata

深圳北理莫斯科大学
УНИВЕРСИТЕТ МГУ–ППИ В ШЭНЬЧЖЭНЕ
SHENZHEN MSU–BIT UNIVERSITY

Table 4: F1 scores of TASTE and the baseline approaches on WikiTable and GitTables datasets when only metadata are used as input ($l = 20$).

Model	Precision	Recall	F1
WikiTable dataset			
TURL w/o content	0.6787	0.5627	0.6153
DODUO w/o content	0.5266	0.6534	0.5832
TASTE w/o P2	0.9037	0.9057	**0.9047**
GitTables dataset			
TURL w/o content	0.9855	0.9753	0.9804
DODUO w/o content	0.9893	0.9832	0.9862
TASTE w/o P2	0.9941	0.9843	**0.9892**

TASTE efficiently utilizes table metadata for semantic type prediction:
The multi-task learning and the asynmetric metadata tower trained with only metadata allows TASTE to be robust in strict privacy setting in the cloud.

Evaluation: Ratio of Scanned Columns

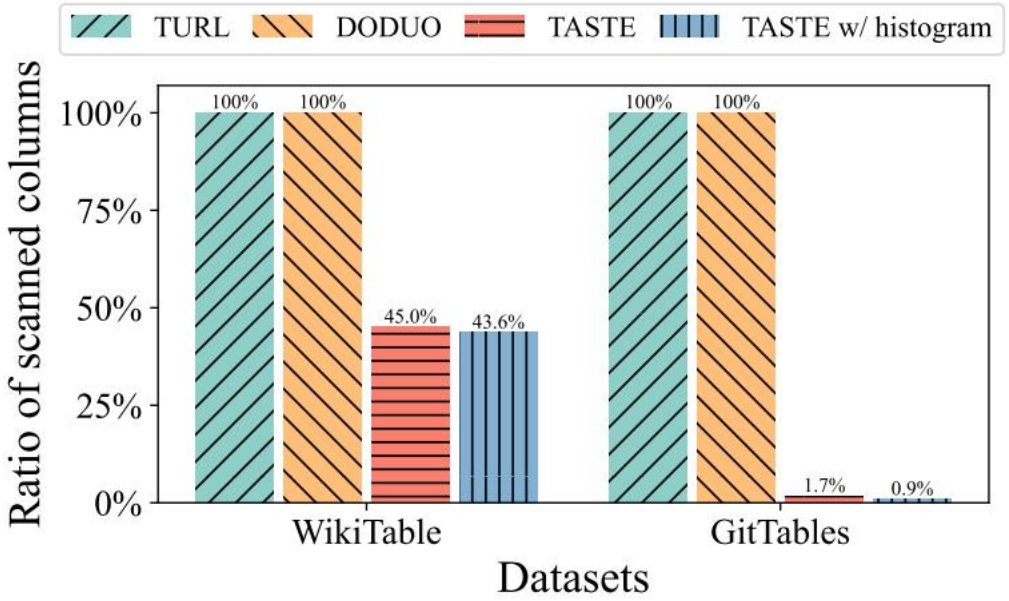


Figure 5: Ratio of scanned columns by different DL-based semantic type detection approaches.

TASTE is much less intrusive to user tables.

Evaluation: When No-Type Columns are Common

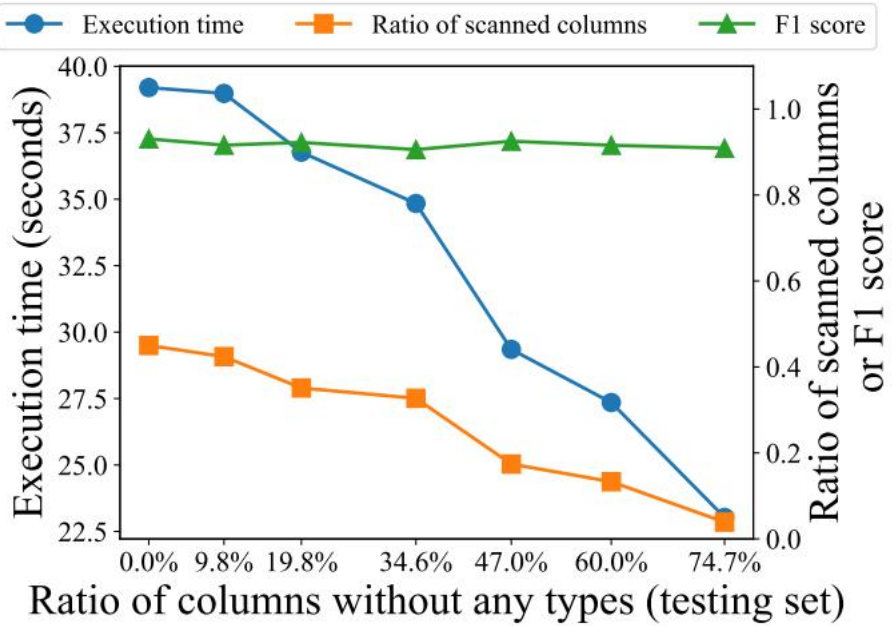
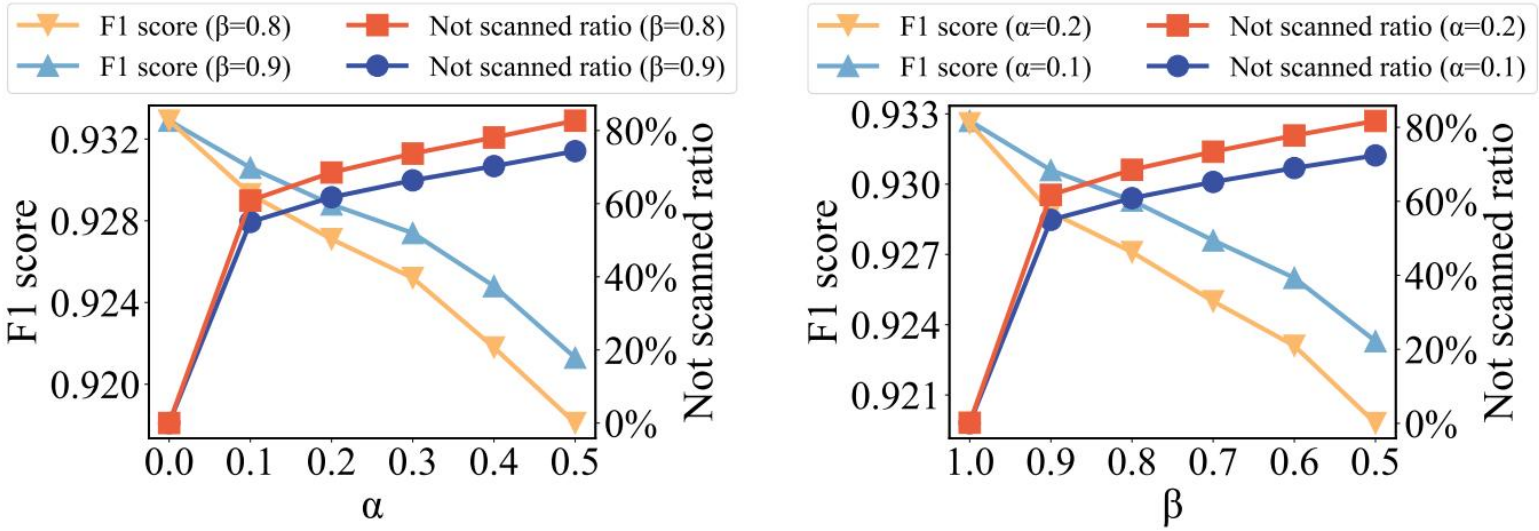


Figure 6: Performance of TASTE when the ratio of columns without any types changes (WikiTable).

TASTE is much more effecient when columns without types are common.

Evaluation: Sensitivity to hyper-parameter settings

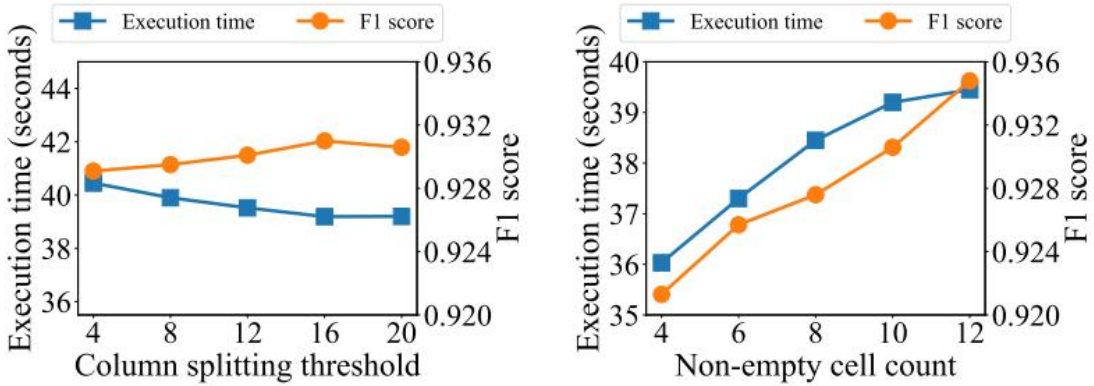


(a) Varying α with fixed β (b) Varying β with fixed α

Figure 7: Effects of varying α and β (WikiTable).

- **α increases, fewer columns are uncertain, less P2, fewer columns scanned, but lower F1.**
- **β increases, more columns are uncertain, more P2, higher F1, but more columns scanned.**

Evaluation: Sensitivity to hyper-parameter settings



(a) Varying l with $n = 10$ (b) Varying n with $l = 20$

Figure 8: The impact of l and n on performance (WikiTable).

We split big tables with more than $l$columns into smaller ones, and read only $n$on-empty cell values to meet the model's input sequence length constraint.

- **If larger $l$for table intactness, better F1 score and execution time performance. But larger $l$requires more GPU resources.**
- **If larger $n$for more information, higher F1 score but longer execution time.**

深圳北理莫斯科大学
УНИВЕРСИТЕТ МГУ–ППИ В ШЭНЬЧЖЭНЕ
SHENZHEN MSU–BIT UNIVERSITY

Thanks

Code: https://github.com/ncols-bytes/taste.git
Homepage: liangfengsid.github.com
Email: fliang@smbu.edu.cn