TAPAS: Weakly Supervised Table Parsing via Pre-training

Jonathan Herzig[1,2], Paweł Krzysztof Nowak[1], Thomas Müller[1], Francesco Piccinno[1], Julian Martin Eisenschlos[1]

[1]Google Research
[2]School of Computer Science, Tel-Aviv University
{jherzig,pawelnow,thomasmueller,piccinno,eisenjulian}@google.com

The contributions of this paper:

- Present TAPAS, an approach to question answering over tables without generating logical forms.
- Extends BERT's architecture to encode tables as input, initializes from an effective joint pre-training of text segments and tables crawled from Wikipedia, and is trained end-to-end.
- https://github.com/google-research/tapas

This is implemented by extending BERT's architecture (Devlin et al., 2019) with additional embeddings that capture tabular structure, and with two classification layers for selecting cells and predicting a corresponding aggregation operator.

↓

- 基于Bert的拓展——额外的embedding
- 两个分类层

millions of tables and related text segments crawled from Wikipedia. During pre-training, the model masks some tokens from the text segment and from the table itself, where the objective is to predict the original masked token based on the textual and tabular context.

↓

- 用Wikipedia的表格和相关文本片段训练
- Mask和Predict的模式

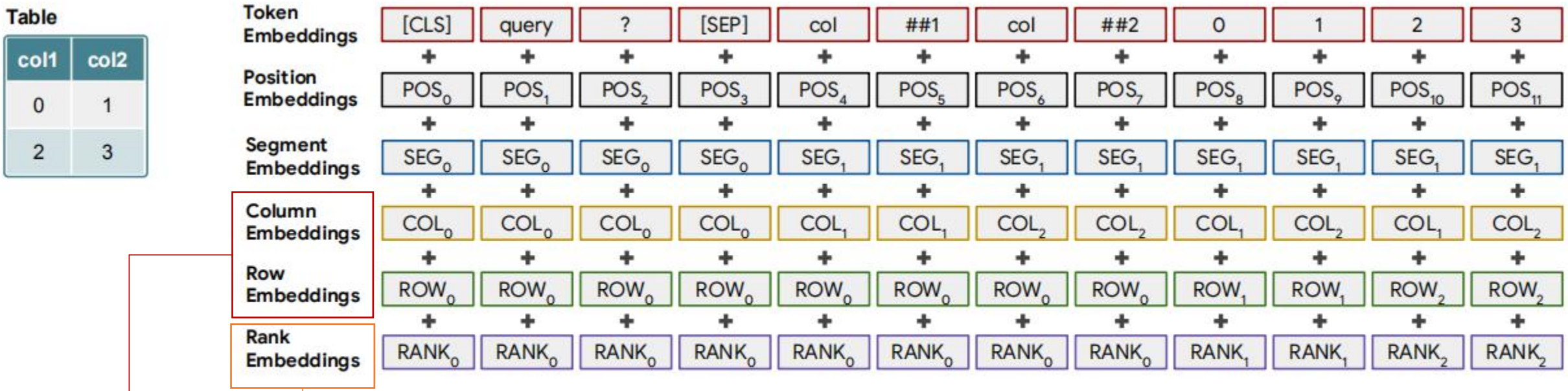
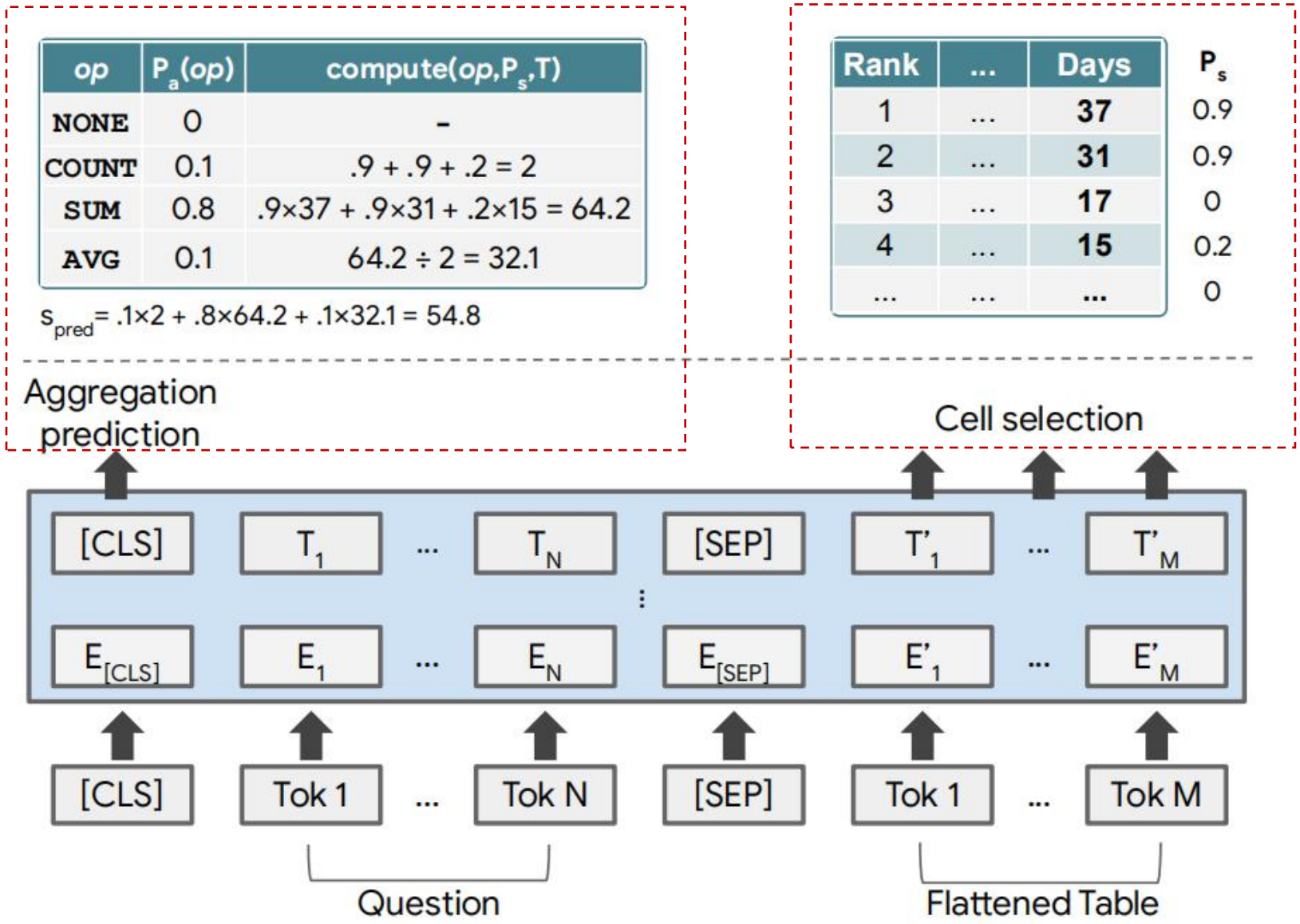


Figure 2: Encoding of the question "query?" and a simple table using the special embeddings of TAPAS. The previous answer embeddings are omitted for brevity.

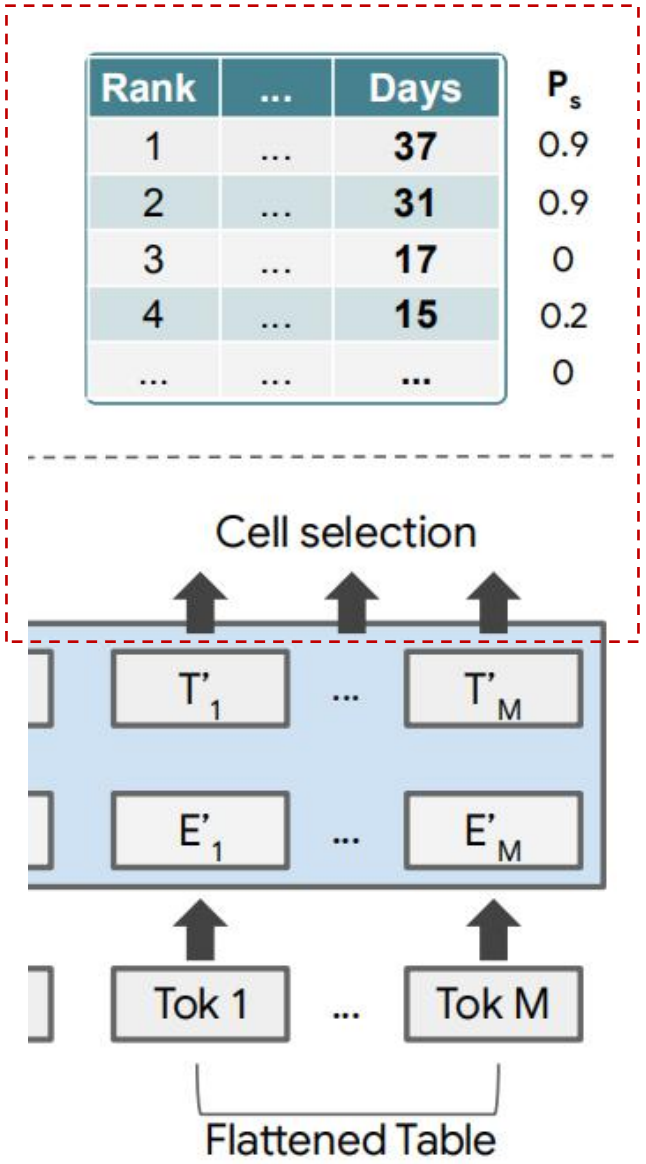
- ***Previous Answer：*** the current question might refer to the previous
- ***Rank ID：*** floats or dates
- ***Column / Row ID：*** is the index of the column/row that this token appears in, or 0



Question: “Total number of days for the top two”.

***classification layer*：**

selects a subset of the table cells



independent **Bernoulli variables**

↓

compute the **logits** { use a linear layer on top of its last hidden vector

↓

average { cell logit = the average over logits of tokens in that cell

↓

softmax

↓

$p_s^{(c)}$ { probability to select cell *c*

another linear layer

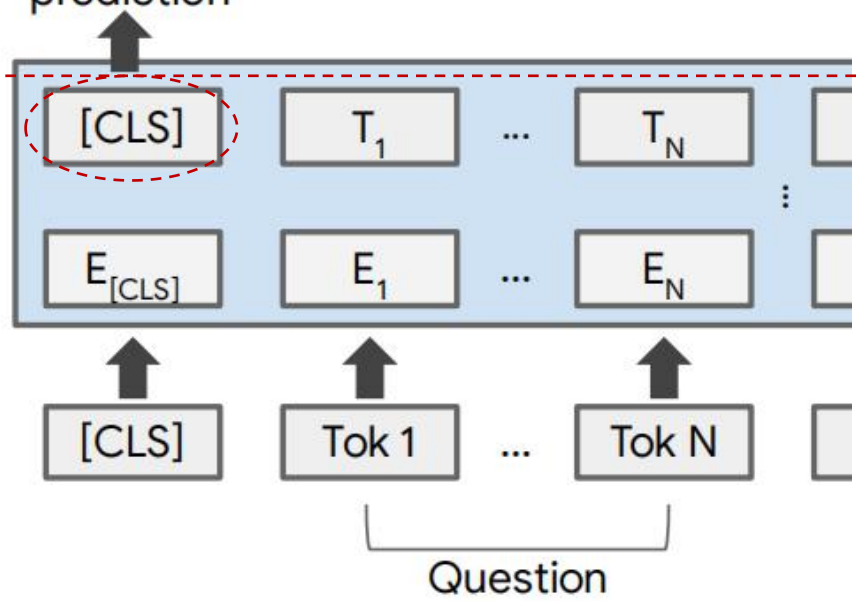
additional logit

classification layer：

select the aggregation operator.

op	$P_a(op)$	compute(op,P_s,T)
NONE	0	-
COUNT	0.1	.9 + .9 + .2 = 2
SUM	0.8	.9×37 + .9×31 + .2×15 = 64.2
AVG	0.1	64.2 ÷ 2 = 32.1

s_{pred} = .1×2 + .8×64.2 + .1×32.1 = 54.8



$p_s^{(c)} > 0.5$

scalar

the table values: (op, p_s, T)

calculate the **expected** result

$$s_{\text{pred}} = \sum_{i=1} \hat{p}_{\text{a}}(op_i) \cdot \text{compute}(op_i, p_{\text{s}}, T),$$

Huber loss (Huber, 1964) given by:

$$\mathcal{J}_{\text{scalar}} = \begin{cases} 0.5 \cdot a^2 & a \leq \delta \\ \delta \cdot a - 0.5 \cdot \delta^2 & \text{otherwise} \end{cases}$$

$$a = |s_{\text{pred}} - s|$$

	WikiSQL	WikiTQ	SQA
Logical Form	✓	✗	✗
Conversational	✗	✗	✓
Aggregation	✓	✓	✗
Examples	80654	22033	17553
Tables	24241	2108	982

Table 2: Dataset statistics.

Model	Dev	Test
Liang et al. (2018)	71.8	72.4
Agarwal et al. (2019)	74.9	74.8
Wang et al. (2019)	79.4	79.3
Min et al. (2019)	84.4	**83.9**
TaPas	**85.1**	83.6
TaPas (fully-supervised)	88.0	86.4

Table 3: WikiSQL denotation accuracy[4].

Model	ALL	SEQ	Q1	Q2	Q3
Pasupat and Liang (2015)	33.2	7.7	51.4	22.2	22.3
Neelakantan et al. (2017)	40.2	11.8	60.0	35.9	25.5
Iyyer et al. (2017)	44.7	12.8	70.4	41.1	23.6
Sun et al. (2018)	45.6	13.2	70.3	42.6	24.8
Müller et al. (2019)	55.1	28.1	67.2	52.7	46.8
TaPas	**67.2**	**40.4**	**78.2**	**66.0**	**59.7**

Table 5: SQA test results. ALL is the average question accuracy, SEQ the sequence accuracy, and QX, the accuracy of the X'th question in a sequence.

Model	Test
Pasupat and Liang (2015)	37.1
Neelakantan et al. (2017)	34.2
Haug et al. (2018)	34.8
Zhang et al. (2017)	43.7
Liang et al. (2018)	43.1
Dasigi et al. (2019)	43.9
Agarwal et al. (2019)	44.1
Wang et al. (2019)	44.5
TaPas	42.6
TaPas (pre-trained on WikiSQL)	48.7
TaPas (pre-trained on SQA)	48.8

Table 4: WikiTQ denotation accuracy.

	SQA (SEQ)		WikiSQL		WikiTQ	
all	39.0		84.7		29.0	
-pos	36.7	-2.3	82.9	-1.8	25.3	-3.7
-ranks	34.4	-4.6	84.1	-0.6	30.7	+1.8
-{cols,rows}	19.6	-19.4	74.1	-10.6	17.3	-11.6
-table pre-training	26.5	-12.5	80.8	-3.9	17.9	-11.1
-aggregation	-		82.6	-2.1	23.1	-5.9

Datasets and Results

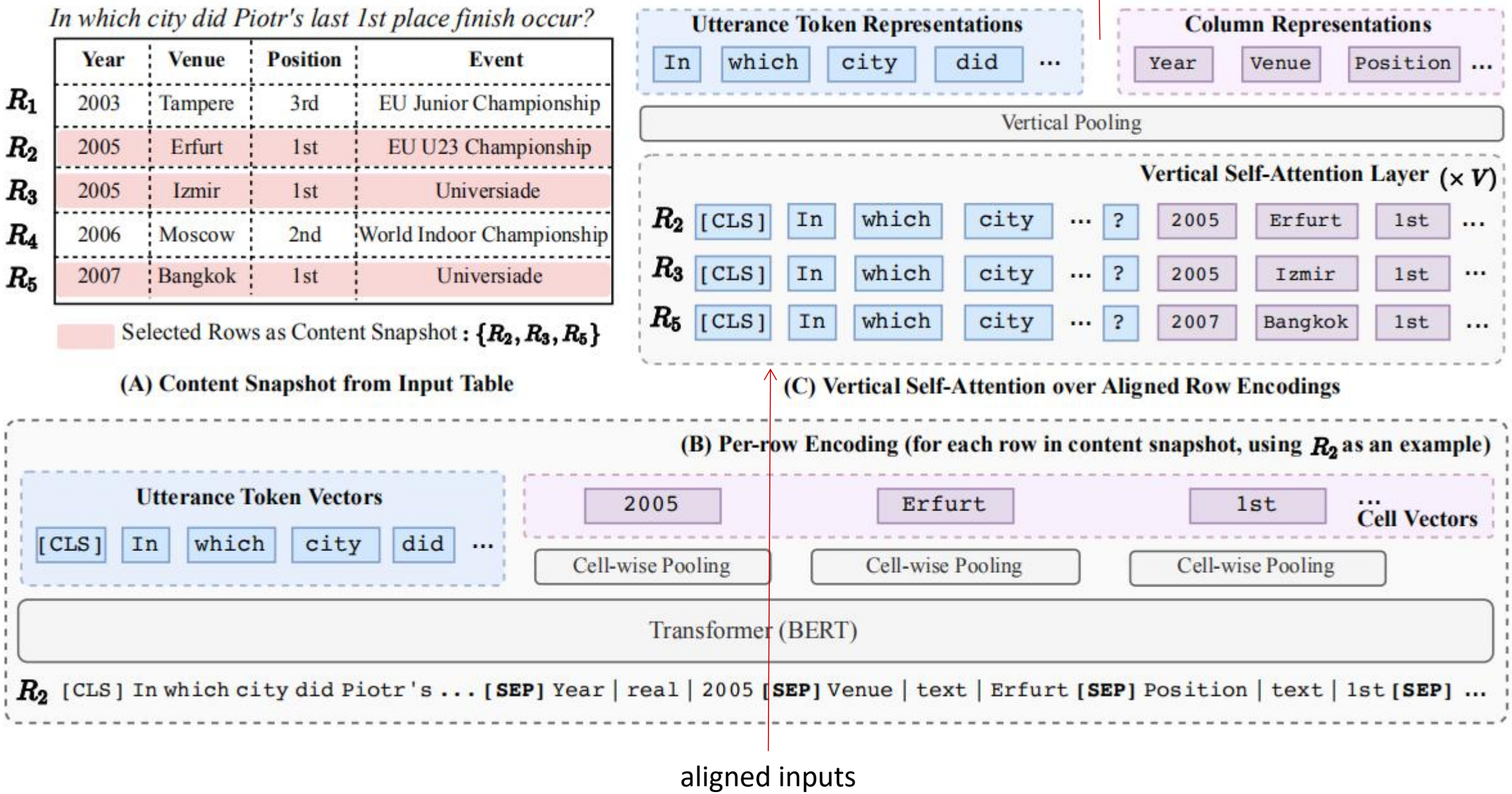
TABERT: Pretraining for Joint Understanding of Textual and Tabular Data

Pengcheng Yin* **Graham Neubig** **Wen-tau Yih** **Sebastian Riedel**
Carnegie Mellon University Facebook AI Research
{pcyin,gneubig}@cs.cmu.edu {scottyih,sriedel}@fb.com

The contributions of this paper:

- Present TABERT, a pretrained LM that jointly learns representations for NL sentences and (semi-)structured tables.
- TABERT is trained on a large corpus of 26 million tables and their English contexts.
- https://github.com/facebookresearch/TaBERT

be used by downstream neural semantic parsers



aligned inputs

Content Snapshot

In which city did Piotr's last 1st place finish occur?

	Year	Venue	Position	Event
R_1	2003	Tampere	3rd	EU Junior Championship
R_2	2005	Erfurt	1st	EU U23 Championship
R_3	2005	Izmir	1st	Universiade
R_4	2006	Moscow	2nd	World Indoor Championship
R_5	2007	Bangkok	1st	Universiade

Selected Rows as Content Snapshot : $\{R_2, R_3, R_5\}$

(A) Content Snapshot from Input Table

K

K >1:
select the top-K rows
the highest n-gram overlap ratio with the utterance

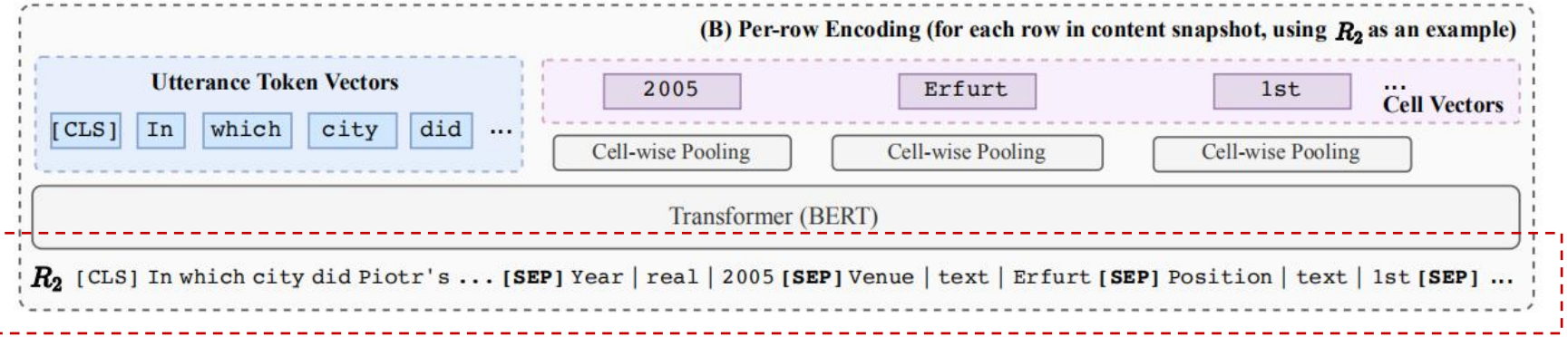
K=1:
create a synthetic row by selecting the cell values from each column that have the highest n-gram overlap with the utterance

eg. "How many more participants were there in 2008 than in the London Olympics?" ,
the most relevant cells to the utterance, 2008 (from Year) and London (from Host City), are from different rows

---

Row Linearization

Year | real | 2005
Column Name | Column Type | Cell Value



- English Wikipedia
- WDC WebTable Corpus

26.6 million parallel examples

Previous Systems on WikiTableQuestions

Model	DEV	TEST
Pasupat and Liang (2015)	37.0	37.1
Neelakantan et al. (2016)	34.1	34.2
Ensemble 15 Models	37.5	37.7
Zhang et al. (2017)	40.6	43.7
Dasigi et al. (2019)	43.1	44.3
Agarwal et al. (2019)	43.2	44.1
Ensemble 10 Models	–	46.9
Wang et al. (2019b)	43.7	44.5

Our System based on MAPO (Liang et al., 2018)

	DEV	Best	TEST	Best
Base Parser†	42.3 ±0.3	42.7	43.1 ±0.5	43.8
w/ $\text{BERT}_{\text{Base}}$ (K = 1)	49.6 ±0.5	50.4	49.4 ±0.5	49.2
– content snapshot	49.1 ±0.6	50.0	48.8 ±0.9	50.2
w/ $\text{TABERT}_{\text{Base}}$ (K = 1)	51.2 ±0.5	51.6	50.4 ±0.5	51.2
– content snapshot	49.9 ±0.4	50.3	49.4 ±0.4	50.0
w/ $\text{TABERT}_{\text{Base}}$ (K = 3)	51.6 ±0.5	52.4	51.4 ±0.3	51.3
w/ $\text{BERT}_{\text{Large}}$ (K = 1)	50.3 ±0.4	50.8	49.6 ±0.5	50.1
w/ $\text{TABERT}_{\text{Large}}$ (K = 1)	51.6 ±1.1	52.7	51.2 ±0.9	51.5
w/ $\text{TABERT}_{\text{Large}}$ (K = 3)	**52.2** ±0.7	**53.0**	**51.8** ±0.6	**52.3**

Table 1: Execution accuracies on WIKITABLEQUESTIONS. †Results from Liang et al. (2018). (TA)BERT models are evaluated with 10 random runs. We report mean, standard deviation and the best results. TEST↦BEST refers to the result from the run with the best performance on DEV. set.

Top-ranked Systems on Spider Leaderboard

Model	DEV. ACC.
Global–GNN (Bogin et al., 2019a)	52.7
EditSQL + BERT (Zhang et al., 2019a)	57.6
RatSQL (Wang et al., 2019a)	60.9
IRNet + BERT (Guo et al., 2019)	60.3
+ Memory + Coarse-to-Fine	61.9
IRNet V2 + BERT	63.9
RyanSQL + BERT (Choi et al., 2020)	**66.6**

Our System based on TranX (Yin and Neubig, 2018)

	Mean	Best
w/ $\text{BERT}_{\text{Base}}$ (K = 1)	61.8 ±0.8	62.4
– content snapshot	59.6 ±0.7	60.3
w/ $\text{TABERT}_{\text{Base}}$ (K = 1)	63.3 ±0.6	64.2
– content snapshot	60.4 ±1.3	61.8
w/ $\text{TABERT}_{\text{Base}}$ (K = 3)	63.3 ±0.7	64.1
w/ $\text{BERT}_{\text{Large}}$ (K = 1)	61.3 ±1.2	62.9
w/ $\text{TABERT}_{\text{Large}}$ (K = 1)	64.0 ±0.4	64.4
w/ $\text{TABERT}_{\text{Large}}$ (K = 3)	**64.5** ±0.6	**65.2**

Table 2: Exact match accuracies on the public development set of SPIDER. Models are evaluated with 5 random runs.

ColNet: Embedding the Semantics of Web Tables for Column Type Prediction

Jiaoyan Chen[1], Ernesto Jiménez-Ruiz[2,4], Ian Horrocks[1,2], Charles Sutton[2,3]
[1]Department of Computer Science, University of Oxford, UK
[2]The Alan Turing Institute, London, UK
[3]School of Informatics, The University of Edinburgh, UK
[4]Department of Informatics, University of Oslo, Norway

The contributions of this paper:

- Propose a neural network based column type annotation framework named ColNet which is able to integrate KB reasoning and lookup with machine learning and can automatically train Convolutional Neural Networks for prediction.

- https://github.com/alan-turing-institute/SemAIDA/tree/master/AAAI19

ColNet(framework) = KB + word representations + machine learning

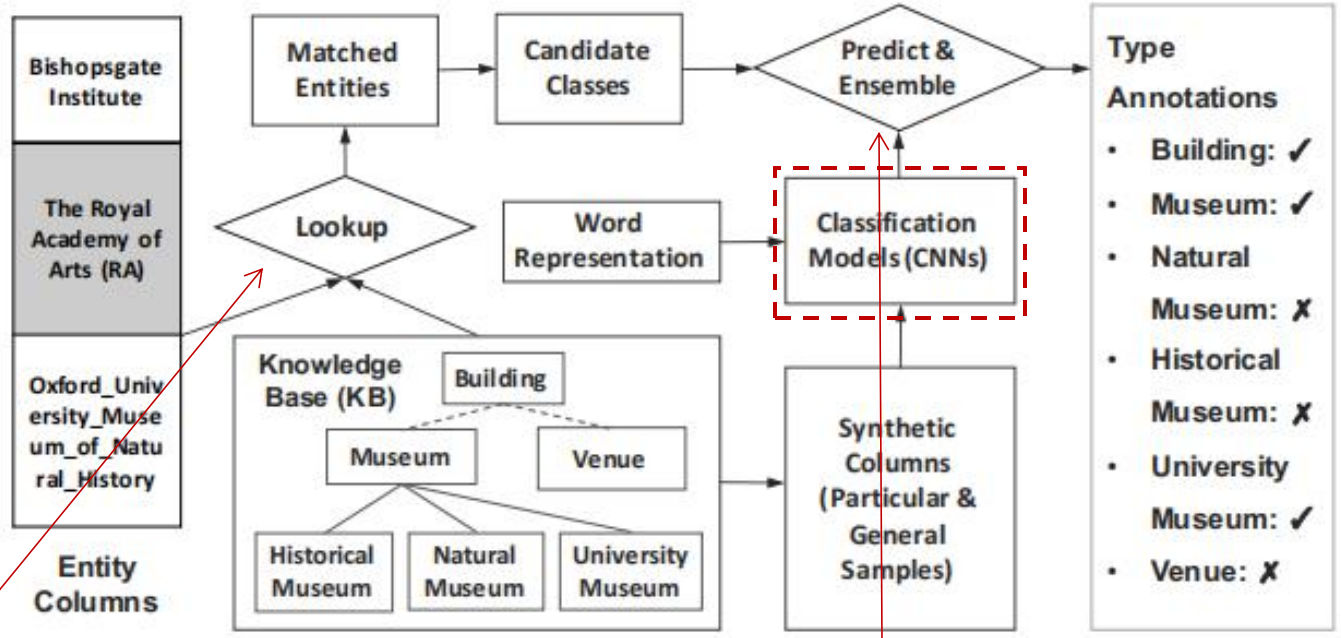


Figure 1: Column Type Annotation with ColNet

Step1: *lookup*

retrieves column cells' corresponding entities candidate classes

labeled samples

Step2: *prediction*

calculates a score

predict

Step3: *ensemble*

recall cells missed by lookup

given a column and a candidate class combines the vote with the score

particular samples: a close data distribution to the column cells

general samples: deal with the challenge of sample shortage

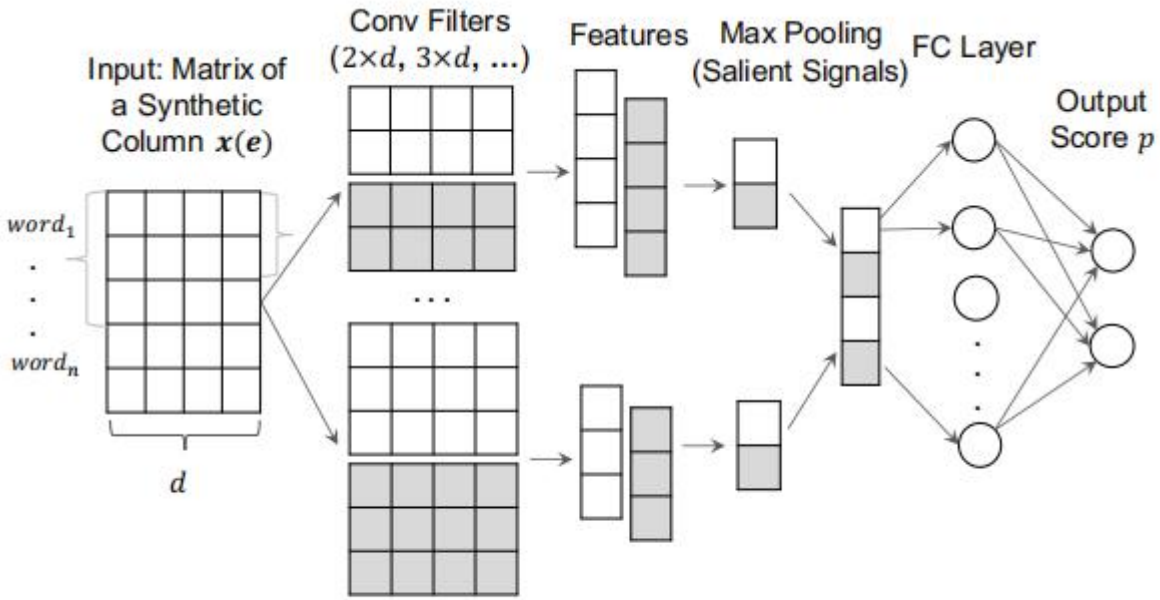


Figure 2: The CNN architecture used in ColNet.

Name	Columns	Avg. Cells	Different "Best" ("Okay") Classes
T2Dv2	411	124	56 (35)
Limaye	428	23	21 (24)

Table 1: Some statistics of the web table sets.

Models	Methods	All Columns	PK Columns
Tolerant	$\text{ColNet}_{\text{Ensemble}}$	0.917, 0.909, 0.913	0.967, 0.985, 0.976
	ColNet	0.845, 0.896, 0.870	0.927, 0.960, 0.943
	Lookup-Vote	0.909, 0.865, 0.886	0.965, 0.960, 0.962
	T2K Match	0.664, 0.773, 0.715	0.738, 0.895, 0.809
Strict	$\text{ColNet}_{\text{Ensemble}}$	0.853, 0.846, 0.849	0.941, 0.958, 0.949
	ColNet	0.765, 0.811, 0.787	0.868, 0.898, 0.882
	Lookup-Vote	0.862, 0.821, 0.841	0.946, 0.941, 0.943
	T2K Match	0.624, 0.727, 0.671	0.729, 0.884, 0.799

Table 2: Results (precision, recall, F1 score) on T2Dv2.

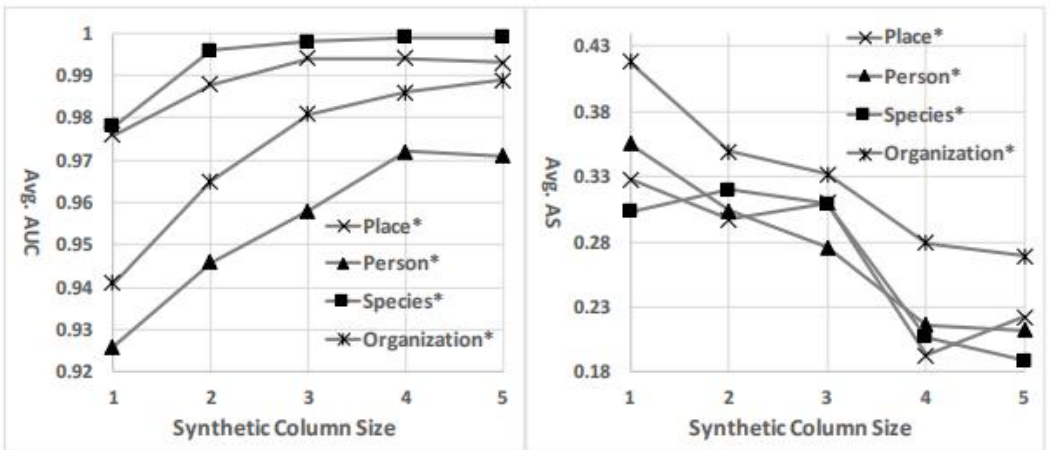


Figure 3: The performance of CNNs on TM classes [left] and FM classes [right] under different synthetic column sizes, trained by particular samples.

Models	Methods	PK Columns
Tolerant	$\text{ColNet}_{\text{Ensemble}}$	0.796, 0.799, 0.798
	ColNet	0.763, 0.820, 0.791
	Lookup-Vote	0.732, 0.660, 0.694
	T2K Match	0.560, 0.408, 0.472
	Efthymiou17-Vote	0.759, 0.414, 0.536
Strict	$\text{ColNet}_{\text{Ensemble}}$	0.602, 0.639, 0.620
	ColNet	0.576, 0.619, 0.597
	Lookup-Vote	0.571, 0.447, 0.501

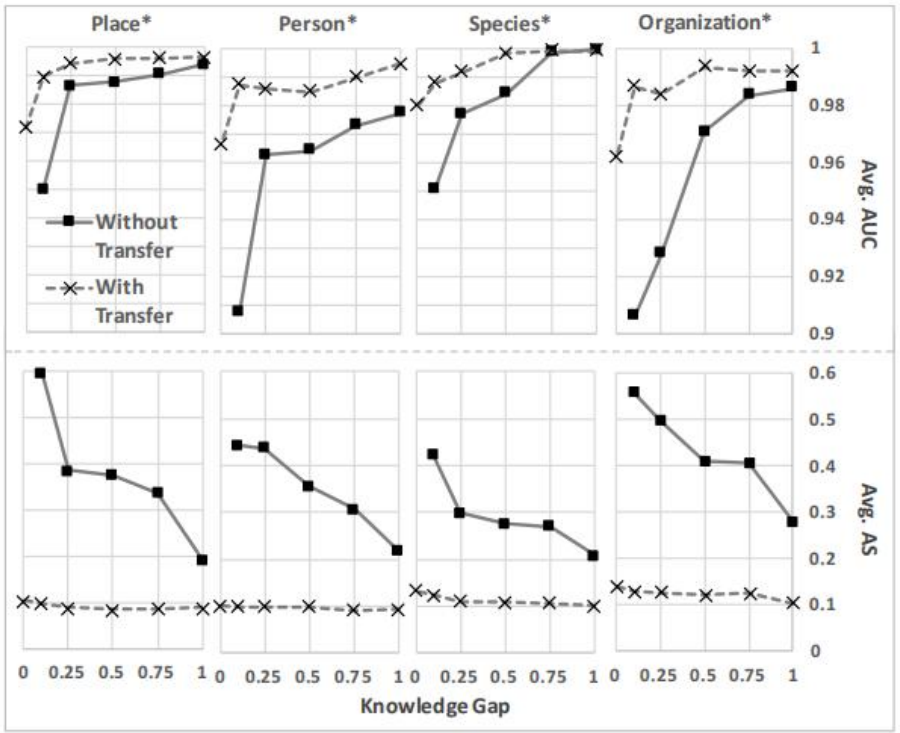


Figure 4: The performance of CNNs of TM classes [above] and FM classes [below], under different knowledge gaps, with and without transfer learning. Knowledge gap is simulated by randomly selecting a ratio of particular entities for training. The lower ratio, the larger gap.

Learning Semantic Annotations for Tabular Data

Jiaoyan Chen[1] , **Ernesto Jiménez-Ruiz**[2,4] , **Ian Horrocks**[1,2] , **Charles Sutton**[2,3]

[1]Department of Computer Science, University of Oxford, UK
[2]The Alan Turing Institute, London, UK
[3]School of Informatics, The University of Edinburgh, UK
[4]Department of Informatics, University of Oslo, Norway

The contributions of this paper:

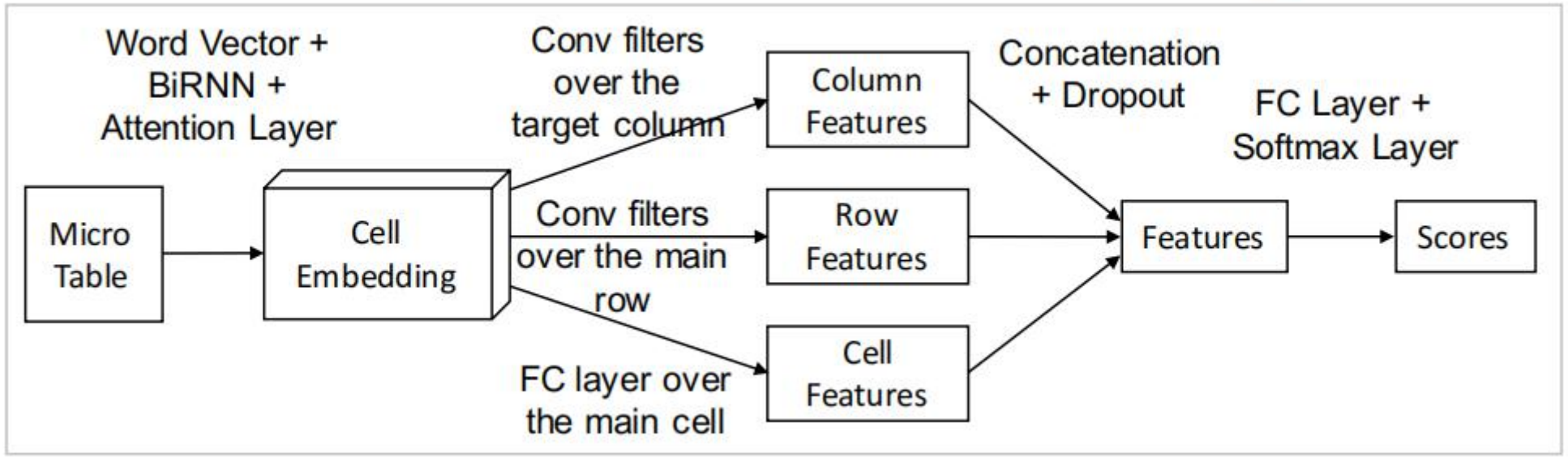
• Propose a deep prediction model that can fully exploit a table's contextual semantics, including [table locality features learned by a Hybrid Neural Network (HNN)], and [inter-column semantics features learned by a knowledge base (KB) lookup and query answering algorithm].

• https://github.com/alan-turing-institute/SemAIDA/tree/master/IJCAI19

HNN Architecture

$$f_i^{c_1,k_1} = g(W_i^{c_1} \otimes \mathcal{L} + b^{c_1})$$

$$f_j^{c_2,k_2} = g(W_j^{c_2} \otimes [\mathcal{L}_1, L_{1,1}, ..., L_{l,1}] + b^{c_2})$$



HNN mainly includes an attentive BiRNN for cell embedding, and a customized convolutional (Conv) layer for table locality feature learning, as shown in Figure 1.

We assume a table is composed of cells organized by columns and rows, without any metadata like column names.

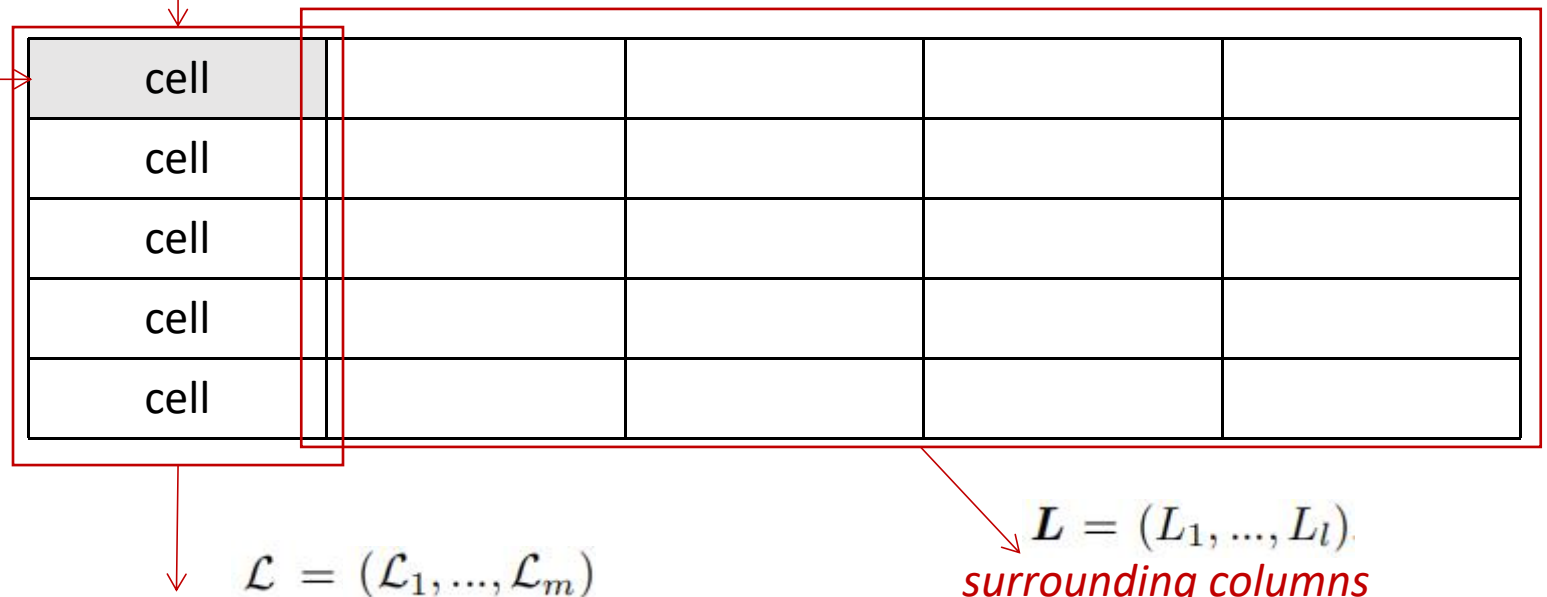
averaged as the final score vector to the target column:

$$\bar{y} = \frac{1}{M-m+1} \sum_{i=1}^{M-m+1} \mathcal{F}(S_i)$$

each cells——

mention entity: a sequence of words

main cell



$$\mathcal{L} = (\mathcal{L}_1, ..., \mathcal{L}_m)$$

target column: whose type is to be predicted

$$\boldsymbol{L} = (L_1, ..., L_l)$$

surrounding columns

We assume a fixed set of candidate classes that are disjoint with each other are given, denoted as $\{C_1, ..., C_K\}$.

The problem is assigning a real value score to each candidate class so that the correct class (type) of the target column has the highest score.

Property Features

use to represent the potential relations between the target column and its surrounding columns

Algorithm 1: P2VecExtract $\langle(\mathcal{L}, \boldsymbol{L}), \boldsymbol{P}, \mathbb{N}, \alpha\rangle$

Input: *(i)* A micro table $(\mathcal{L}, \boldsymbol{L})$, *(ii)* candidate properties $\boldsymbol{P}$ with the size of d_1, *(iii)* a maximum number of matched entities $\mathbb{N}$, *(iv)* a text matching threshold α,
Result: v: a property vector of the micro table
begin
  $v := zeros(d_1)$; % *Init. of the property vector*
  $E := entity\_lookup(\mathcal{L}_1, \alpha)$; % *Entity lookup by main cell*
  foreach *entity* $e \in E$ **do**
    $T := query(e)$;% *Get triples whose subject is* e
    foreach *triple* $(s, p, o) \in T$ *with* $p \in \boldsymbol{P}$ **do**
      foreach *surrounding column* $L_i \in \boldsymbol{L}$ **do**
        if $cell\_object\_match(L_{i,1}, o, \alpha)$ **then**
          $j := index(p, \boldsymbol{P})$;
          $v[j] := 1$; % *Set the slot of the property*
  $v := v / \|v\|$; % *Normalization*
return v

现有一个知识库M，该知识库是基于描述逻辑的。则***M=<Tbox,Abox>***。

TBox：是有关概念和关系的蕴涵断言集合，描述概念和关系的一般属性。其定义了特定知识领域的结构并包含一系列公理，可以通过已有概念构成新的概念。——>***RDF-Schema***

ABox：是有关个体的实例断言集合，断言一个个体是某个概念的实例，或者两个个体之间存在某种关系。是一个描述关于具体个体事实的公理集，包含的是外延性知识。——>***SPARQL***

The ABox contains entities, each of which is represented by an URI (Uniform Resource Identifier), and RDF triples $\langle s, p, o\rangle$, where s represents a subject (an entity), p represents a predicate (a property) and o represents an object (either an entity or a data value like date and number). An entity can belong to one or more classes, which is defined by the property *rdf:type*.

Methods	T2D-Te	Limaye	Efthymiou
word2vec-avg + FC-Softmax	0.925	0.561	0.582
word2vec-avg + CNN^{c}	**0.947**	0.597	**0.619**
word2vec-avg + CNN^{r}	0.872	**0.675**	0.460
word2vec-avg + CNN^{cr}	0.902	0.667	0.531
Att-BiRNN	0.955	0.597	0.648
Att-BiRNN + CNN^{c}	**0.962**	0.632	**0.655**
Att-BiRNN + CNN^{r}	0.880	0.684	0.529
Att-BiRNN + CNN^{cr}	0.925	**0.728**	0.581

Table 1: Accuracy of HNN variants. word2vec-avg represents averaging the word2vec of words of each cell phrase. FC-Softmax denotes a classifier by a FC layer and a Softmax layer. The superscripts c and r of CNN denote Conv filters over the column and row.

Methods	T2D-Te	Limaye	Efthymiou
P2Vec	0.939	**0.759**	0.609
HNN	**0.962**	0.728	**0.655**
Ensemble I (P2Vec + HNN)	**0.966**	0.697	0.629
Ensemble II (P2Vec + HNN)	0.959	**0.746**	**0.650**

Table 2: Accuracy of P2Vec, HNN, and the ensemble approaches. Both LR and MLP are used and the average is reported.

Thanks