UNDERSTANDING OF SEMI-STRUCTURED INFORMATION

COV884
Rohit Katariya

AGENDA

EMBDI: Creating Embeddings of Heterogeneous Relational Datasets

TAPAS: Weakly Supervised Table Parsing via Pre-training

Reviews

TABERT: Pretraining for Joint Understanding of Textual and Tabular Data

RPT: Relational Pre-trained Transformer Is Almost All You Needtowards Democratizing Data Preparation

References

EMBDI: CREATING EMBEDDINGS OF HETEROGENEOUS RELATIONAL DATASETS

Riccardo Cappuzzo — EURECOM
Paolo Papotti — EURECOM
Saravanan — QCRI, HBKU

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

REVIEWS

TAPAS: WEAKLY SUPERVISED TABLE PARSING VIA PRE-TRAINING

TABERT: PRETRAINING FOR JOINT UNDERSTANDING OF TEXTUAL AND TABULAR DATA

Pengcheng Yin, Graham Neubig — Carnegie Mellon University
Wen-tau Yih, Sebastian Riedel — Facebook AI Research

RPT:RELATIONAL PRE-TRAINED TRANSFORMER IS ALMOST ALL YOU NEED TOWARDS DEMOCRATIZING DATA PREPARATION

Nan Tang — QCRI, HBKU
Ju Fan — Renmin University
Fangyi Li et al. — Renmin University

REFERENCES

EMBDI: CREATING EMBEDDINGS OF HETEROGENEOUS RELATIONAL DATASETS

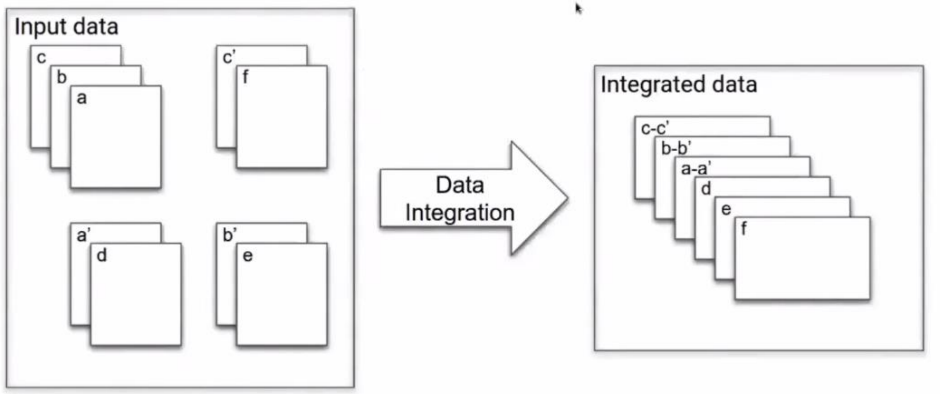
Riccardo Cappuzzo
EURECOM

Paolo Papotti
EURECOM

Saravanan
QCRI, HBKU

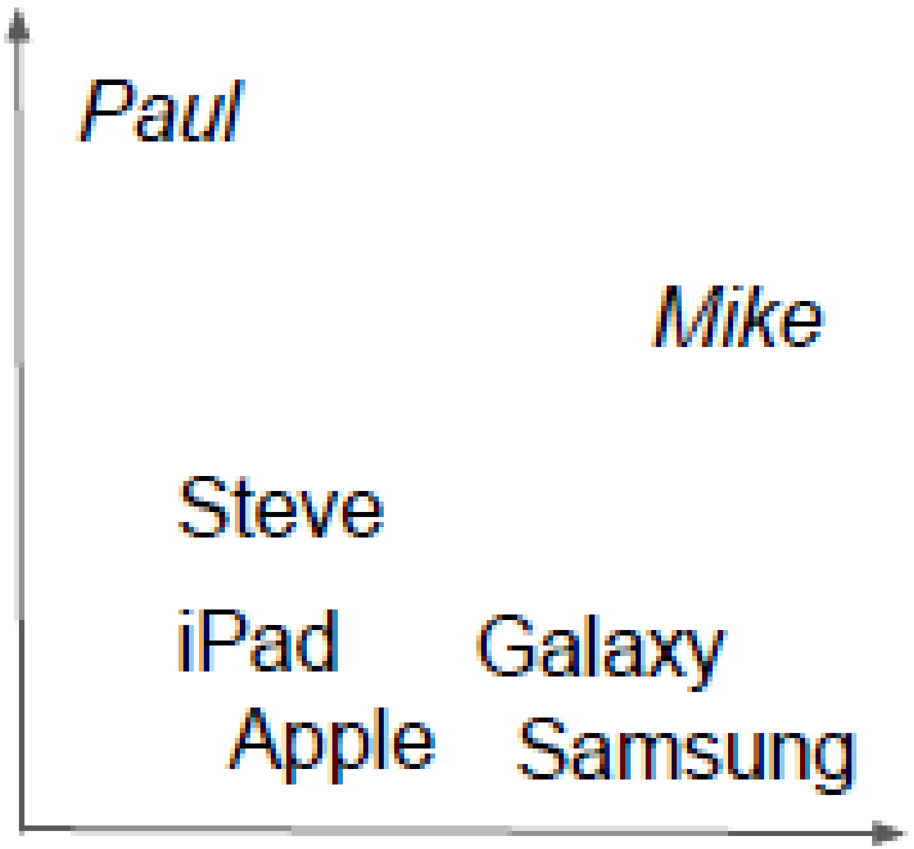
EMBDI: CREATING EMBEDDINGS OF HETEROGENEOUS RELATIONAL DATASETS

- Unsupervised Data Integration through Local Embeddings for Relational Databases
- Relationship specification through graph-based representation
- Data Integration
  - Schema Matching
  - Entity resolution
  - Token Matching

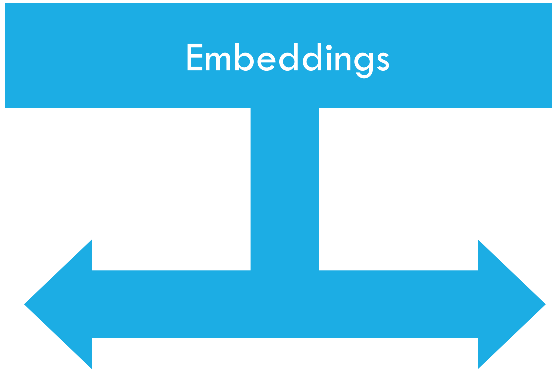


EMBDI:EMBEDDINGS

- Embeddings: Represent words/ sentences/ more general entities
- Vector Spaces: Geometric properties of categorical data; numerical representation; distances between different points



EMBDI: EMBEDDINGS



Pretrained

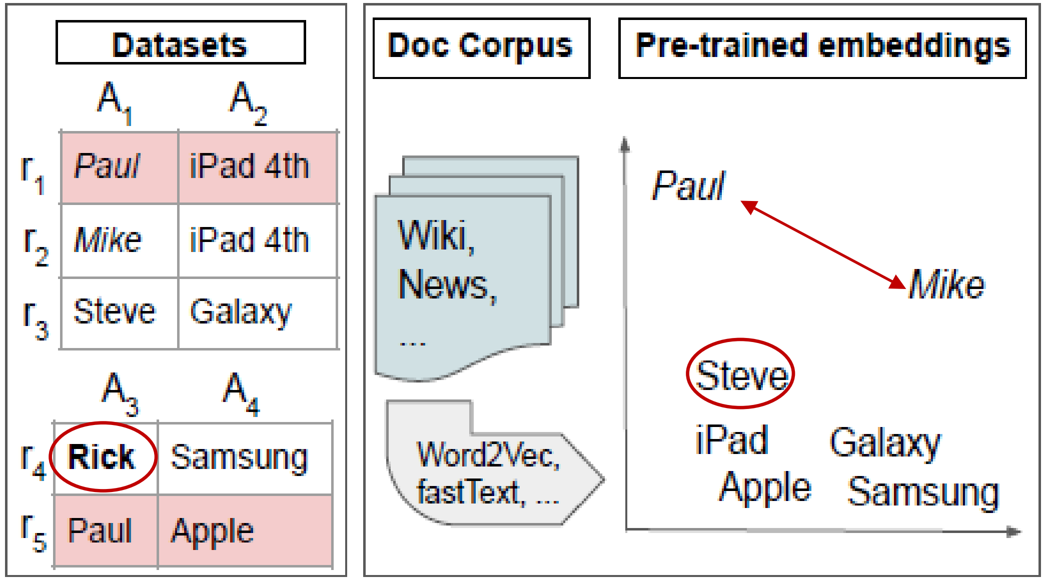
Locally Trained (Specific to Dataset)

Don’t capture

- OOV words
- Dataset Semantics
- Noisy background information

No trivial way

- How to encode relational semantics
- Limited information
- Different number of tuples
- Incomplete and noisy



EMBDI: LOCAL EMBEDDINGS FOR DATA INTEGRATION

Tuples are not Sentences

- Ignores semantics of relational data
  - Eg Mike is related to every item in its column, row.
- No natural ordering of attributes.
- Databases are normalized to remove redundancy
- Hierarchical relationship of data (cells/tuples/attributes/dataset)

Embeddings should span multiple datasets

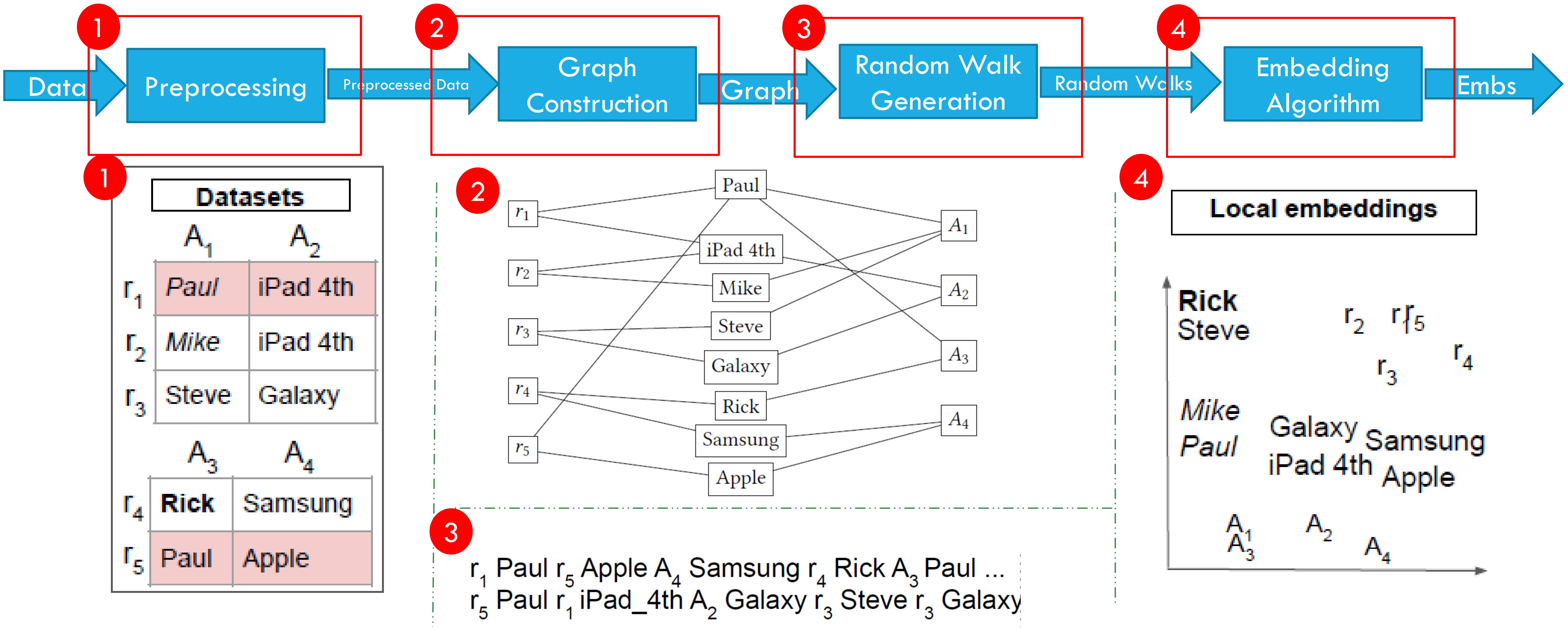
- All datasets don't have same attributes
- Must be able to leverage similarity across multiple datasets
  - Tuple-tuple (r1-r5)
  - Attribute-Attribute (A1-A3)

Datasets

	A_1	A_2
r_1	*Paul*	iPad 4th
r_2	*Mike*	iPad 4th
r_3	Steve	Galaxy

	A_3	A_4
r_4	**Rick**	Samsung
r_5	Paul	Apple

EMBDI: FRAMEWORK



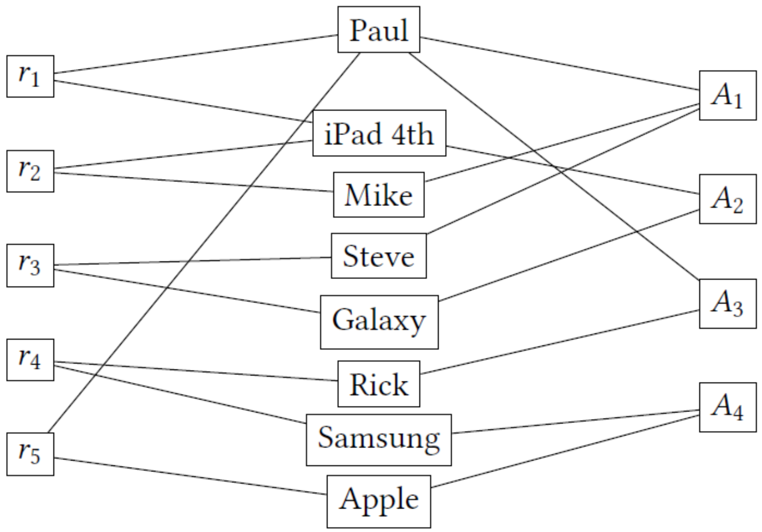
EMBDI: GRAPH CONSTRUCTION

Naïve Approach:

- Treat tuples as sentence
  - Looses relationships
- Complete subgraph
  - $\binom{n}{k}$ edges per tuple
  - Ignores token1, token2 belong to same column

EmbDI Approach:

- Compact tripartite graph
- Encodes inherent relationships
- Incorporate external information like tokens are synonyms of each other
- Unified view of multiple datasets



EMBDI: GRAPH CONSTRUCTION

- Representation:
  - Token Nodes
  - Record Id Nodes (RIDs)
  - Column Id Nodes(CIDs)
- Synonym merging:
  - Domain information
  - Using external sources like wordnet
- Numerical Values:
  - Rounded to number of significant digits
  - Treated as regular nodes
  - Data Distribution aware distances between numbers

Algorithm 1 GenerateTripartiteGraph

Input: relational dataset D
let G = empty graph
for all c_i in columns(D) **do**
  G.addNode(c_i)
for all r_i in rows(D) **do**
  G.addNode(R_i) //R_i is the record id of r_i
  for all value v_k in r_i **do**
    if v_k is multi-word **then**
      for all word in tokenize(v_k) **do**
        G.addNode(word)
        G.addEdge(word, R_i), G.addEdge(word, c_k)
    else if v_k is single-word **then**
      G.addNode(v_k)
      G.addEdge(v_k, R_i), G.addEdge(v_k, c_k)
Output: graph G

Key advantage

- Same expressive power as the complete sub-graph
- Requiring orders of magnitude fewer edges

EMBDI: SENTENCE CONSTRUCTION

- Graph embeddings generation problem
- Random walks to quantify the similarity between neighboring nodes
- Exploit metadata like tuple and attribute ids
  - Nodes with similar neighborhood => close in final embeddings
- Agnostic to type of random walks used

Type Agnostic Random Walks

Different choices yielding different embeddings
- Biased towards nodes belonging to same tuples
- Biased towards rare nodes

Tuple and Attribute Embedding

- Inherent tuple and attribute embds
- No need to use averaging of tokens

Budgeting

Assign a “budget” each node to guarantee all nodes will be the starting point of at least “budget” random walks

Uniform Size Random Walks

- Guarantee good execution times on large datasets
- Provide high quality results

Algorithm 2 GenerateRandomWalk

Input: starting node n_j, random walk length l
r_j = findNeighboringRID(n_j)
W = seq(r_j, n_j)
currentNode = n_j
while length(W) < l **do**
    nextNode = findRandomNeighbor(currentNode)
    W.add(nextNode)
    currentNode = nextNode
Output: walk W

EMBDI: EMBEDDING CONSTRUCTION

- Generated sentences are then pooled together to build training corpus
- Agnostic to word embedding algorithms
  - GloVe
  - Word2vec
  - fastText
  - Other newer embedding training algorithm

Algorithm 3 Meta Algorithm for EmbDI

Input: relational datasets D, number of random walks n_{walks}, number of nodes n_{nodes}
$W = []$
G = GenerateTripartiteGraph(D)
for all $n_j \in nodes(G)$ **do**
    for i = 1 to (n_{walks}/n_{nodes}) **do**
        w_i = GenerateRandomWalk(n_j)
        W.add(w_i)
E = GenerateEmbeddings(W)
Output: Local relational embeddings E

EMBDI: IMPROVING EMBEDDINGS

- **Node Imbalance**: Different number of nodes in different datasets
  - Effective heuristic :start random walks from nodes in both datasets
- **Overlapping nodes**: Bridges to 2 datasets
  - Start with attributes nodes that are common to 2 datasets.
- **Handling missing, noisy data**
  - Imputation techniques – expensive, not generic
  - Single node for all null values vs Multiple nodes for null values
  - Placeholders for columns
- **Node Merging:** External tables, matchers based on syntactic similarity, pretrained embeddings, clusters
- **Node Replacement:** Only when we are confident about 2 nodes refer to same entity. While sentence creation T_i replaced with T_j with confidence 0.8.

EMBDI: EMBEDDING ALIGNMENT

- Embeddings for multiple relations
    a) Training embeddings one relation at a time
    b) Pooling relations and training a common space
- a is is more scalable but misses out on patterns
- b: larger relations do not overpower smaller ones

Embedding alignment

- Changing the vector space of one dataset to better match the vector space of the other.

Algorithm 4 AlignEmbeddings

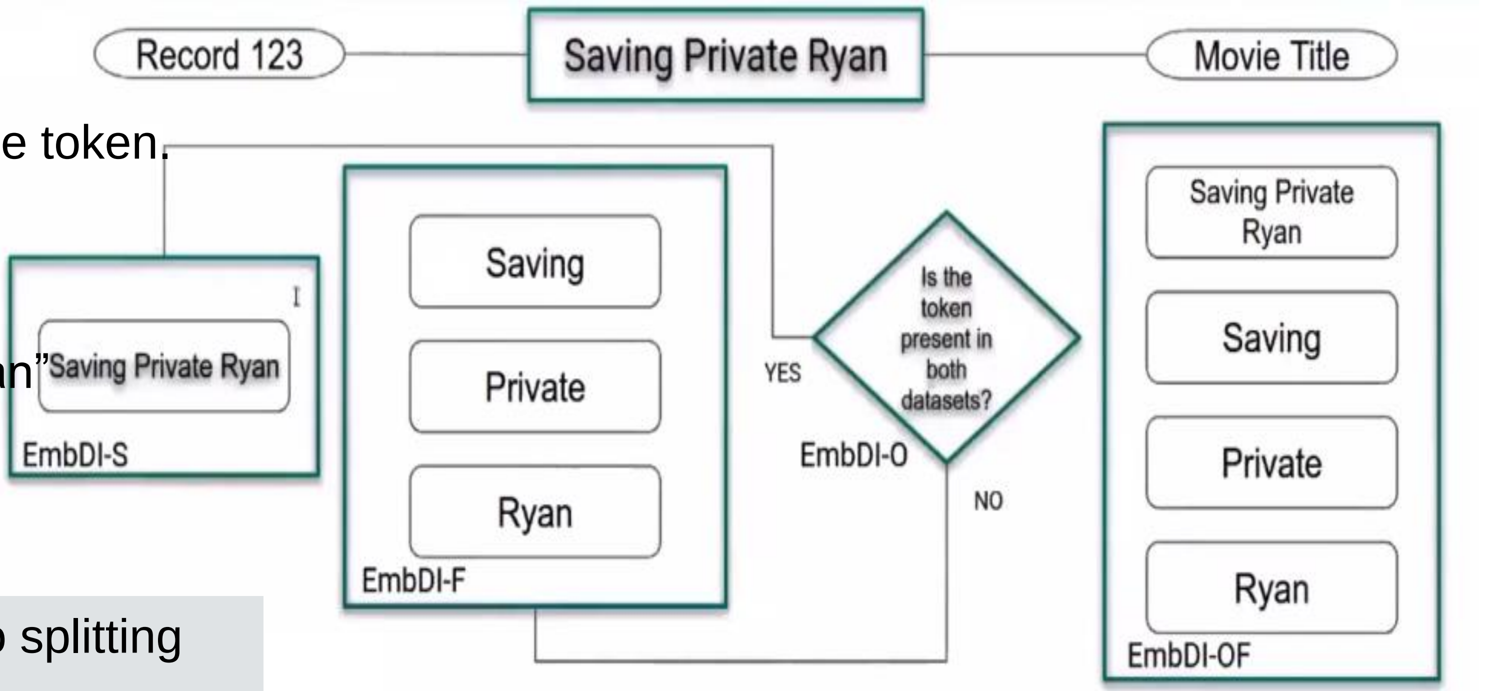
Input: relations $\mathbb{R}_1, \mathbb{R}_2, \mathbb{E} = \text{EmbDI}(\text{concat}(\mathbb{R}_1, \mathbb{R}_2))$
let U_i be the set of unique words in $\mathbb{R}_i\ \forall i \in 1, 2$
let $\mathcal{A} = U_1 \cap U_2$
$A = \mathbb{E}(w_i)\ \forall\ w_i \in \mathbb{R}_1$
$B = \mathbb{E}(w_j)\ \forall\ w_j \in \mathbb{R}_2$
$W^* = \text{argmin}_{W, \mathcal{A}}(WA - B)$
$A' = W^* A$
for all $w_i \in \mathbb{R}_1 \cup \mathbb{R}_2$ **do**
  if $w_i \in \mathbb{R}_1 \cap \mathbb{R}_2$ **then**
    $\mathbb{E}'(w_i) = \text{average}(A'(w_i), B(w_i))$
  else if $w_i \in \mathbb{R}_1$ **then**
    $\mathbb{E}'(w_i) = A'(w_i)$
  else
    $\mathbb{E}'(w_i) = B(w_i)$
Output: Aligned embeddings $\mathbb{E}'$

Key advantage

- Better materialize relationships between tokens
- Geometric relationships between tokens within each individual dataset are retained

EMBDI: HANDLING MULTI-WORD TOKENS

- Multiword tokens
  a) Entire word sequence as a single token.
  b) tokenize the word sequence
- No concrete answer
  - Eg “Adobe” and “Photoshop”
  - Movie name “Saving Private Ryan”



Heuristic

- Nodes that are present in datasets: no splitting
- Only in one dataset: split
- Helps in preserving bridges between datasets, also identifies the logical entities

Embeddings Configurations

- EmbDI-S
- EmbDI-F
- EmbDI-O
- EmbDI-OF

EMBDI: SCHEMA MATCHING

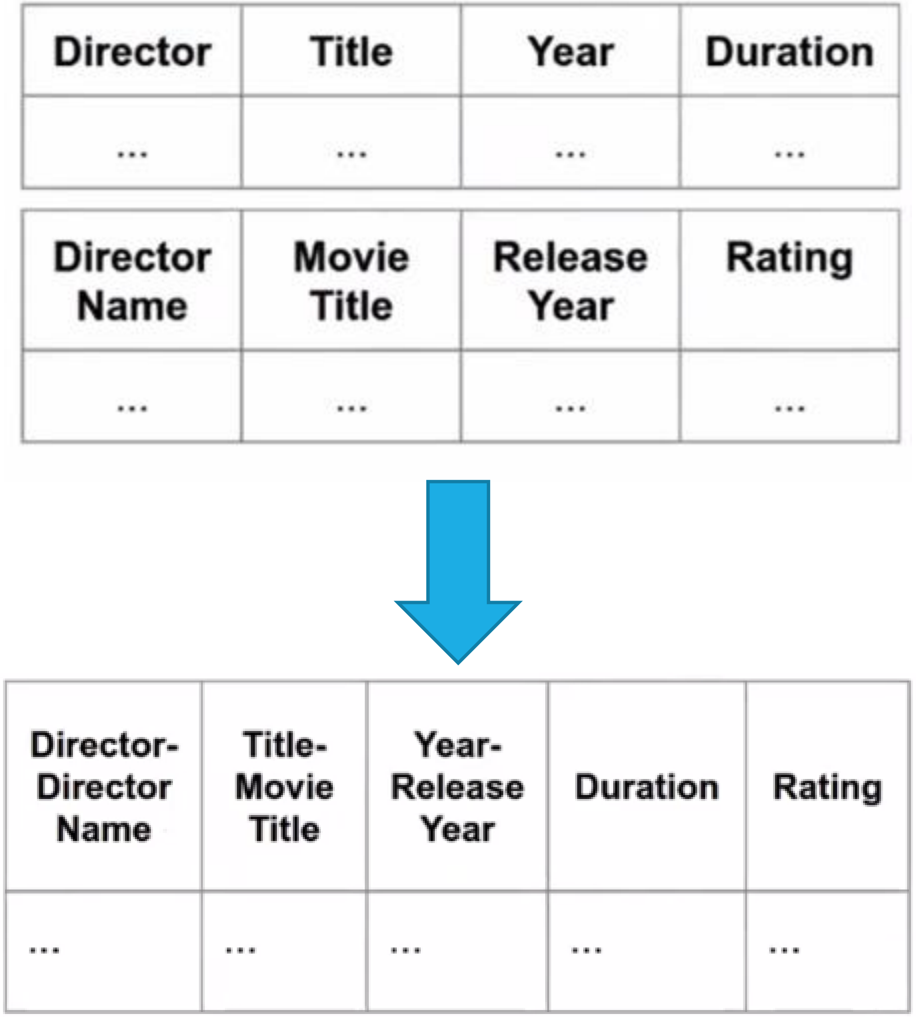
Aims to build new schema which combines columns from different datasets.

Traditional approaches

- based on the value distributions
- other similarity measures
- both syntactic and semantic similarities
- embeddings only on attribute/relation names

EmbDI approach:

- on attribute vectors themselves
- exploiting their cosine distance in the vector space
- prevent false positives: terminate after two iterations



EMBDI: SCHEMA MATCHING

Aims to build new schema which combines columns from different datasets.

Traditional approaches

- based on the value distributions
- other similarity measures
- both syntactic and semantic similarities
- embeddings only on attribute/relation names

EmbDI approach:

- on attribute vectors themselves
- exploiting their cosine distance in the vector space
- prevent false positives: terminate after two iterations

Algorithm 5 Schema Matching

let C_1 be the set of CIDs of dataset D_1 and C_2 be the set of CIDs of dataset D_2
let $d(c_i)$ be the list of distances between column $c_i \in C_1$ and all other columns $c_k \in C_2$, sorted in ascending order of distance (and viceversa).
let $\mathcal{T} = C_1 \cup C_1$ be the set of columns to be matched
while $\mathcal{T} \neq \emptyset$ **do**
    for all $c_k \in \mathcal{T}$ **do**
        if $d(c_k) \neq \emptyset$ **then**
            c'_k = findClosest($d(c_k)$)
            c''_k = findClosest($d(c'_k)$)
            if $c''_k == c_k$ **then**
                c_k and c'_k are matched
                remove c_k, c'_k from $\mathcal{T}$
            else
                removeCandidate($d(c_k)$, c'_k)
                removeCandidate($d(c'_k)$, c_k)
        else
            remove c_k from $\mathcal{T}$

EMBDI: ENTITY RESOLUTION

Traditional approaches

- Combination of methods over tuple terms
  - averaging embeddings
  - concatenating embeddings

EmbDI approach:

- use of RIDs as nodes in the heterogenous graph
- unsupervised ER by computing the distance between RIDs

Algorithm 6 Entity Resolution

let $\mathcal{R}_1$ be the set of RIDs $\in D_1$
let $\mathcal{R}_2$ be the set of RIDs $\in D_2$
let $d(r_i)$ be the list of distances between RID $r_i \in \mathcal{R}_i$ and the closest n_{top} RIDs $\in D_j$, with $i \neq j$.
for all $r_i \in D_1 \cup D_2$ **do**
    $d(r_i) = \text{findClosest}(r_i, n_{top})$
for all $r_k \in D_1$ **do**
    $r'_k = \text{findClosest}(d(r_k))$
    $r''_k = \text{findClosest}(d(r'_k))$
    if $r''_k == r_k$ **then**
        r_k and r'_k are matched

EMBDI: TOKEN MATCHING

- Matching tokens that are conceptual synonyms of each other
- String matching
- Eg: “English” while other could encode it as “EN”
- Different from schema matching, not identifying attributes that represent the same information
- Additional signal to be combined with the other similarity measures
  - e.g., edit distance, Jaccard,TF/IDF

Algorithm:

1. Input: relations A_i, A_j, t_k = IdentifySynonym ($t_k \in Dom(A_i)$)
2. For token t_k , identify the set of top-n token ids that are closest to t_k
3. first token $t_l \in Dom(A_j)$ is synonym of t_k

EMBDI: DATASETS

Name (shorthand)	# tuples	# columns	# distinct values	# matches	# sentences	% overlap
IMDB-Movielens (IM)	49875	15	118779	4115	2810900	8.79
Amazon-Google (AG)	4589	3	5390	1166	166316	6.01
Walmart-Amazon (WA)	24628	5	45454	961	1168033	3.10
Itunes-Amazon (IA)	62830	8	53079	131	1931816	5.84
Fodors-Zagats (FZ)	864	6	3282	109	69100	9.08
DBLP-ACM (DA)	4910	7	6555	2223	191083	62.33
DBLP-Scholar (DS)	66879	4	131099	5346	3299633	2.33
BeerAdvo-RateBeer (BB)	7345	4	11260	67	310083	10.18
Million Songs Dataset (MSD)	1000000	5	870841	1292023	31180683	n.a.

EMBDI:EMBEDDING EVALUTAION

Embedding Generation Algorithms

Algorithms	Description
BASIC	Creates embeddings from permutations of row tokens and attribute tokens
Pretrained	FastText pretrained embeddings
Node2Vec	widely used algorithm for learning node representation on graphs
HARP	embeddings algo for graph nodes by preserving higher order structural features

Evaluating Embeddings Quality

- MatchAttribute (MA)
  - (Rambo III, The matrix, E.T., A star is born, **M. Douglas**)
- MatchRow (MR)
  - (S. Stallone, Rambo III, **1952**, P. MacDonald)
- MatchConcept (MC)
  - (Q.Tarantino, Pulp fiction, Kill Bill, Jackie Brown, **Titanic**).

EMBDI: EMBEDDING GENERATION ALGORITHMS

	BASIC				NODE2VEC				HARP				EMBDI			
	MA	MR	MC	AVG	MA	MR	MC	AVG	MA	MR	MC	AVG	MA	MR	MC	AVG
BB	**.99**	.33	.32	.55	.97	**.66**	.92	**.85**	.96	.65	**.95**	**.85**	.92	.50	.77	.73
WA	.19	.27	.12	.19	mem	mem	mem	mem	.16	.32	.13	.20	**.94**	**1.00**	**.99**	**.98**
AG	**1.00**	**.42**	.10	.51	**1.00**	.39	**1.00**	**.80**	.99	.37	**1.00**	.79	**1.00**	.38	**1.00**	.79
FZ	.08	.30	.00	.13	.84	.88	.62	.78	.80	.86	.89	.85	**.94**	**.99**	**.94**	**.95**
IA	.09	.11	.09	.09	mem	mem	mem	mem	.81	.59	.96	.78	**.89**	**.85**	**.98**	**.90**
DA	.08	.29	.02	.13	**.79**	.77	.18	.58	.51	.74	.49	.58	**.79**	**.91**	**.66**	**.79**
DS	**1.00**	.58	.69	.76	mem	mem	mem	mem	.12	.06	.06	.08	**.90**	**.99**	**.99**	**.96**
IM	**.99**	.34	.64	**.66**	mem	mem	mem	mem	.07	.29	.10	.16	.74	**.42**	**.78**	.65
MSD	.31	.37	.51	.39	mem	mem	mem	mem	t.o.	t.o.	t.o.	t.o.	**.60**	**.95**	**.83**	**.79**

Fraction of passed tests

EMBDI: SCHEMA MATCHING

	Unsupervised					
	BASE	EMBDI	NODE2VEC	HARP	$SEEP_P$	$SEEP_L$
BB	**1.00**	**1.00**	**1.00**	**1.00**	.75	.75
WA	**1.00**	**1.00**	mem	.60	.60	.80
AG	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
FZ	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
IA	**1.00**	**1.00**	mem	**1.00**	.50	.75
DA	**1.00**	**1.00**	mem	.50	.75	.81
DS	**1.00**	.50	mem	**1.00**	.60	.73
IM	.60	**.78**	mem	**.78**	.68	.75

F Measure results for Schema Matching

EMBDI: ENTITY RESOLUTION

	Unsupervised						Supervised (5% labelled)		Task specific (5% labelled)	
	Pre-trained	Local								
	FASTTEXT	EMBDI-S	EMBDI-F	EMBDI-O	NODE2VEC	HARP	DEEPER$_P$	DEEPER$_L$	DEEPER$_P$	DEEPER$_L$
BB	.59	.50	.82	**.86**	**.86**	**.86**	0.51	0.53	0.54	0.58
WA	.58	.59	.75	**.81**	mem	.78	0.58	0.62	0.62	0.63
AG	.18	.14	.57	.59	.70	**.71**	0.53	0.56	0.58	0.62
FZ	.99	.98	.99	.99	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
IA	.10	.09	.09	.11	mem	.14	.76	.81	.77	**0.82**
DA	.72	.95	.94	.95	.87	**.97**	.84	.89	.86	.90
DS	.80	.85	.75	**.92**	mem	.81	.80	.87	.82	.91
IM	.31	.90	.64	.94	mem	**.95**	.82	.88	.84	.91

F Measure results for Entity Resolution

EMBDI: ABLATION ANALYSIS, TIME ANALYSIS AND FUTURE WORK

Ablation Analysis

- CBOW performs better than Skip-Gram on the ER task, while having worse results in the EQ and SM.
- Decreasing the size of the walks to 5 for the SM task raises the F-measure

Ex

DATASET	Graph	EmbDI Walks	EmbDI Training	Total EmbDI	Node2Vec	HARP
Amazon-Google	1.19	34.36	122.03	156.40	953.3	**135.0**
Beer	2.47	66.65	133.39	**200.04**	1663.4	732.0
DBLP-ACM	2.08	43.64	130.07	173.71	920.5	**128.0**
DBLP-Scholar	33.86	919.81	3027.68	**3947.49**	na	21659.4
Fodors-Zagats	0.28	11.96	40.67	52.63	178.1	**27.0**
IMDB-Movielens	31.56	768.75	2772.17	**3540.93**	na	8001.0
Itunes-Amazon	31.96	533.16	1360.12	**1893.28**	na	9122.0
Walmart-Amazon	13.37	329.10	1113.49	**1442.59**	na	2394.0
MSD	146.05	6377.15	27050.08	**33427.23**	na	na

Future Work

- Combining pre-trained and local embeddings
- Contextual information into word embeddings and language modeling(BERT)

TAPAS: WEAKLY SUPERVISED TABLE PARSING VIA PRE-TRAINING

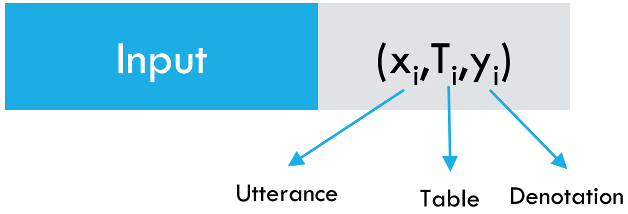
Jonathan Herzig[1,2], Paweł Krzysztof Nowak[1], Thomas Muller[1], Francesco Piccinno[1], Julian Martin Eisenschlos[1]

[1]Google Research, [2]School of Computer Science, Tel-Aviv University

TAPAS: WEAKLY SUPERVISED TABLE PARSING VIA PRE-TRAINING

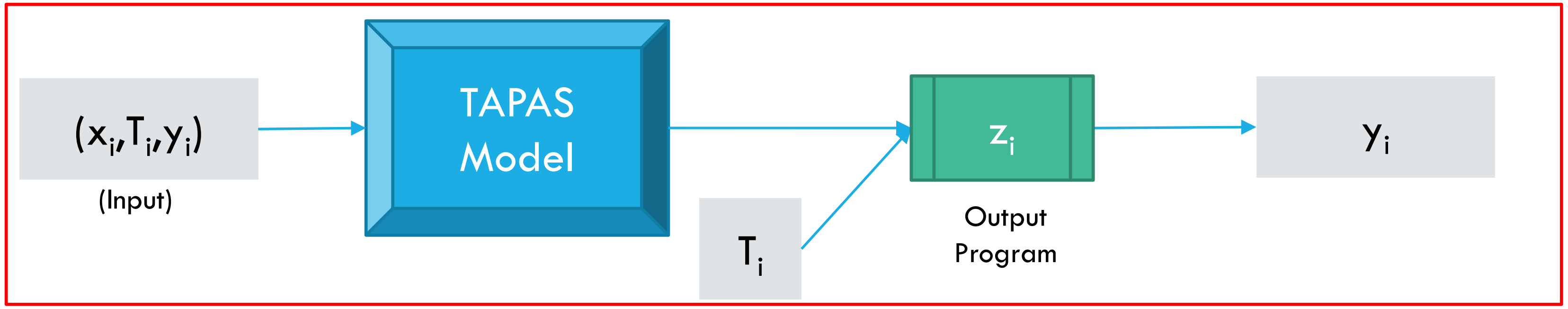
Problem:

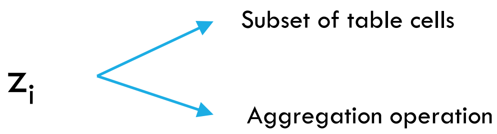
Question answering on semi structured tables, where the model tries to predict a program to be executed on a set of cells to get the answer.



Task

Learn a model that transforms x_i to a program z_i which when applied on T_i gives y_i.





Jonathan Herzig and Pawel Krzysztof Nowak and Thomas Müller and Francesco Piccinno and Julian Martin Eisenschlos (2020). TAPAS: Weakly Supervised Table Parsing via Pre-training. *CoRR, abs/2004.02349*.

TAPAS: SAMPLE TABLE ENCODING

Table

col1	col2
0	1
2	3

Bert Encoder + Positional Embeddings for Tabular Structure

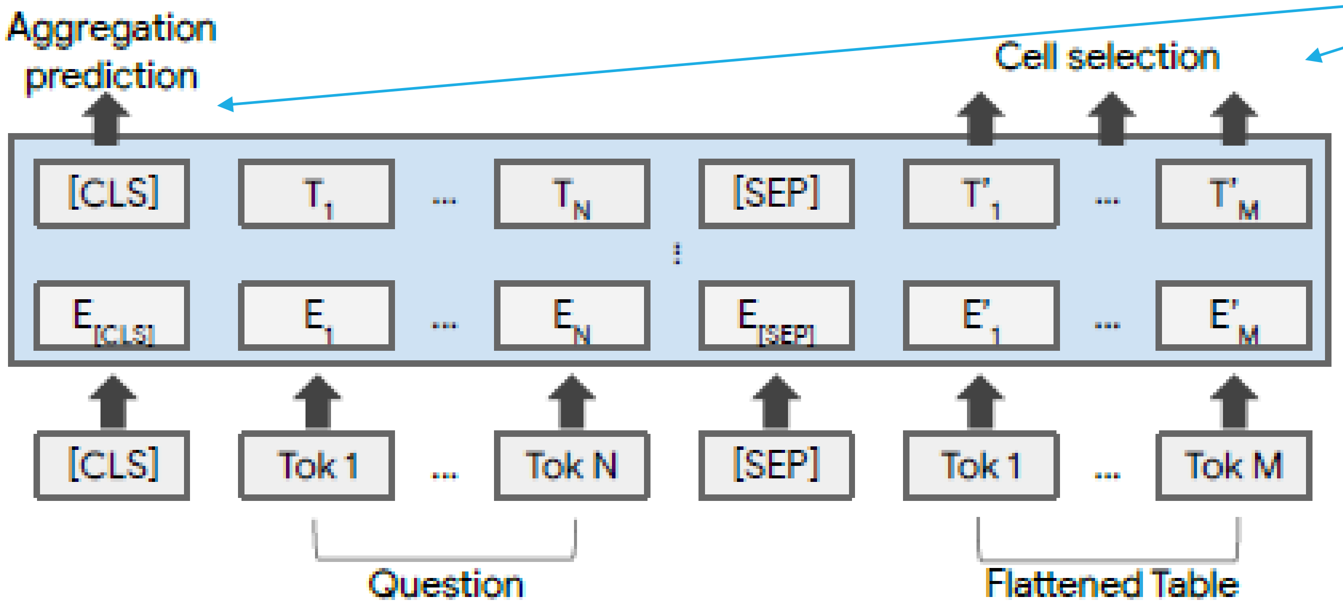
Token Embeddings	[CLS]	query	?	[SEP]	col	##1	col	##2	0	1	2	3	Flattened sequence of words
Position Embeddings	POS_0	POS_1	POS_2	POS_3	POS_4	POS_5	POS_6	POS_7	POS_8	POS_9	POS_{10}	POS_{11}	Positionembeddimgs
Segment Embeddings	SEG_0	SEG_0	SEG_0	SEG_0	SEG_1	SEG_1	SEG_1	SEG_1	SEG_1	SEG_1	SEG_1	SEG_1	Query/table
Column Embeddings	COL_0	COL_0	COL_0	COL_0	COL_1	COL_1	COL_2	COL_2	COL_1	COL_2	COL_1	COL_2	Column Number
Row Embeddings	ROW_0	ROW_0	ROW_0	ROW_0	ROW_0	ROW_0	ROW_0	ROW_0	ROW_1	ROW_1	ROW_2	ROW_2	Row Number
Rank Embeddings	$RANK_0$	$RANK_0$	$RANK_0$	$RANK_0$	$RANK_0$	$RANK_0$	$RANK_0$	$RANK_0$	$RANK_1$	$RANK_1$	$RANK_2$	$RANK_2$	Value's rank in Sorted column

Jonathan Herzig and Pawel Krzysztof Nowak and Thomas Müller and Francesco Piccinno and Julian Martin Eisenschlos (2020). TAPAS: Weakly Supervised Table Parsing via Pre-training. *CoRR, abs/2004.02349*.

TAPAS: MODEL

Classification layers

SUM
COUNT
AVERAGE
NONE



Bias to select cells within a column (Column Logits)

Model Output	•Aggregation operator •Subset of the cells

Jonathan Herzig and Pawel Krzysztof Nowak and Thomas Müller and Francesco Piccinno and Julian Martin Eisenschlos (2020). TAPAS: Weakly Supervised Table Parsing via Pre-training. *CoRR, abs/2004.02349*.

TAPAS: PRE-TRAINING

- Pretraining on tables from Wikipedia
- 6.2M tables ; 3.3M Infobox ; 2.9M WikiTable
- Questions:
  - Table Caption
  - Article/segment's text or description
- Masked language model: Masks some tokens from the text segment and table
  - Whole word masking
  - Whole cell masking
- Predict the original masked tokens based on the textual and tabular context

Jonathan Herzig and Pawel Krzysztof Nowak and Thomas Müller and Francesco Piccinno and Julian Martin Eisenschlos (2020). TAPAS: Weakly Supervised Table Parsing via Pre-training. *CoRR, abs/2004.02349*.

TAPAS: EXAMPLE QUESTIONS

Table

Rank	Name	No. of reigns	Combined days
1	Lou Thesz	3	3,749
2	Ric Flair	8	3,103
3	Harley Race	7	1,799
4	Dory Funk Jr.	1	1,563
5	Dan Severn	2	1,559
6	Gene Kiniski	1	1,131

Example questions

#	Question	Answer	Example Type
1	*Which wrestler had the most number of reigns?*	Ric Flair	Cell selection
2	*Average time as champion for top 2 wrestlers?*	AVG(3749,3103)=3426	Scalar answer
3	*How many world champions are there with only one reign?*	COUNT(Dory Funk Jr., Gene Kiniski)=2	Ambiguous answer
4	*What is the number of reigns for* Harley Race?	7	Ambiguous answer
5	*Which of the following wrestlers were ranked in the bottom 3?*	{Dory Funk Jr., Dan Severn, Gene Kiniski}	Cell selection
	Out of these, who had more than one reign?	Dan Severn	Cell selection

TAPAS: FINETUNING-CELL SELECTION

- Bias to select cells within a column (Column Logits):
  - Model is trained to select a column first
  - Cells to be selected are a subset of this column
- Loss
  - Average cross entropy loss over column selection($J_{columns}$)
  - Average binary cross entropy loss over column cell selections (J_{cells})
  - Aggregation loss (J_{agg})

$$\mathcal{J}_{\text{columns}} = \frac{1}{|\text{Columns}|} \sum_{\text{co} \in \text{Columns}} \text{CE}(p_{\text{col}}^{(\text{co})}, \mathbb{1}_{\text{co}=\text{col}})$$

$$\mathcal{J}_{\text{cells}} = \frac{1}{|\text{Cells}(\text{col})|} \sum_{c \in \text{Cells}(\text{col})} \text{CE}(p_{\text{s}}^{(c)}, \mathbb{1}_{c \in C})$$

$$\mathcal{J}_{\text{aggr}} = -\log p_{\text{a}}(op_0).$$

- Total Loss

$$\boxed{J_{CS} = J_{columns} + J_{cells} + \alpha\, J_{agg}}$$

Hyperparameter

TAPAS: FINETUNING-SCALAR ANSWER

Expected Result

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

Probability distribution
(aggregation operator)

op	$\text{compute}(op, p_{\text{s}}, T)$
COUNT	$\sum_{c \in T} p_{\text{s}}^{(c)}$
SUM	$\sum_{c \in T} p_{\text{s}}^{(c)} \cdot T[c]$
AVERAGE	$\frac{\text{compute}(\text{SUM}, p_{\text{s}}, T)}{\text{compute}(\text{COUNT}, p_{\text{s}}, T)}$

Huber Loss

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

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

Aggregation Loss

$$\mathcal{J}_{\text{aggr}} = -\log(\sum_{i=1} p_{\text{a}}(op_i))$$

Total Loss

$$\mathcal{J}_{\text{SA}} = \mathcal{J}_{\text{aggr}} + \beta \mathcal{J}_{\text{scalar}}$$

Jonathan Herzig and Pawel Krzysztof Nowak and Thomas Müller and Francesco Piccinno and Julian Martin Eisenschlos (2020). TAPAS: Weakly Supervised Table Parsing via Pre-training. *CoRR*, *abs/2004.02349*.

TAPAS: RESULTS

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.

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.

Jonathan Herzig and Pawel Krzysztof Nowak and Thomas Müller and Francesco Piccinno and Julian Martin Eisenschlos (2020). TAPAS: Weakly Supervised Table Parsing via Pre-training. *CoRR, abs/2004.02349*.

TAPAS: SAMPLE RUN

```
Training or predicting ...
Evaluation finished after training step 0.
```

Pos	Player	Team	Span	Innings	Runs	Highest Score	Average	Strike Rate
1	Sachin Tendulkar	India	1989-2012	452	18426	200	44.83	86.23
2	Kumar Sangakkara	Sri Lanka	2000-2015	380	14234	169	41.98	78.86
3	Ricky Ponting	Australia	1995-2012	365	13704	164	42.03	80.39
4	Sanath Jayasuriya	Sri Lanka	1989-2011	433	13430	189	32.36	91.2
5	Mahela Jayawardene	Sri Lanka	1998-2015	418	12650	144	33.37	78.96
6	Virat Kohli	India	2008-2020	236	11867	183	59.85	93.39
7	Inzamam-ul-Haq	Pakistan	1991-2007	350	11739	137	39.52	74.24
8	Jacques Kallis	South Africa	1996-2014	314	11579	139	44.36	72.89
9	Saurav Ganguly	India	1992-2007	300	11363	183	41.02	73.7
10	Rahul Dravid	India	1996-2011	318	10889	153	39.16	71.24

```
> what were the players names?
Sachin Tendulkar, Rahul Dravid, Jacques Kallis, Saurav Ganguly, Inzamam-ul-Haq, Sanath Jayasuriya, Ricky Ponting, Virat Kohli, Mahela Jayawardene, Kuma
> of these, which team did Sachin Tendulkar play for?
India
> what is his highest score?
200
> how many runs has Virat Kohli scored?
11867
```

TAPAS: WEAKLY SUPERVISED TABLE PARSING VIA PRE-TRAINING

Pros

- Simple architecture
- Pretraining
- More question types

Cons

- Fails to capture very large tables
- Fails on multiple aggregations (count of rows with value column1>n)
- Aggregations with multiple tables as context

Jonathan Herzig and Pawel Krzysztof Nowak and Thomas Müller and Francesco Piccinno and Julian Martin Eisenschlos (2020). TAPAS: Weakly Supervised Table Parsing via Pre-training. *CoRR, abs/2004.02349*.

REVIEWS

TAPAS: WEAKLY SUPERVISED TABLE PARSING VIA PRE-TRAINING

STRENGTHS

Both cell selection & aggregation operation can be handled by the model.	Rocktim
Different training objectives were tried.	Rocktim
Intuitive encoding scheme. Simple Architecture	[Da;Har]man
Making the compute() function differentiable - Most "deep-learning" way	Daman
Simple model architecture	Shivangi
Model is learning through the aggregation loss	Shivangi
Ablation study- every newly added positional embedding is an important part of the model	Shivangi
Hierarchical and Table-aware position embeddings	Seshank
Ambiguous category - using current model adds stability	Vishal
Pre-training model - significant impact	Vishal
Model's decisions are interpretable- Contradicting to Daman's point	Harman
Aggregation operation prediction layer that estimates the scalar answer uses the probability distribution ; (Conradiction: Jai)	Shreya; Jai

WEAKNESSES

Fitting to the target word-piece limit;	Rocktim;
Aggregation steps are limited to the operation defined in the paper.	Rocktim
Generalization to larger tables	Daman
Uninterpretable model results; difficult debugging	Daman
Model assigns the highest weight to the correct aggregation function – pit fall	Shivangi
Complicated aggregators like median or mode	Vishal
Column bias unintutive (Contradiction to Aditya,Jai)	Harman; Aditya;Jai
Large model	Harman
For calculating the aggregation operator, the model takes a softmax over the hidden layer vector of the CLS token. It is not very clear to me why this particular token only. Maybe the paper could have talked more about the motivation to choose this and not something else.	Jai

QUESTIONS

Selecting single column for a query seems to be peculiar to the datasets. Can we extend this to multiple columns?	Vishal
Why Infobox tables are beneficial for end tasks?	Vishal
Does token length of 128 limit makes sense when tables which are extracted contain upto 500 cells? How are these snippets created from the tables is not very clear?	Vishal
Modelling of the position embedding here may not be ideal. If we permute the rows of the table, will that affect my model performance? Note that position embedding are important as shown in ablations.	Vishal

EXTENSIONS

Handle large tables or multiple tables - longformers or other novel encoding methods	Rocktim;; Vishal B
Word-piece instead of adding words on a first come first serve basis	Rocktim
Finding a way to generalize the aggregation step to incorporate more operations	Rocktim
Templates for creating NL questions and use that for pre-training	Daman
Use logical forms for supervision	Daman
Data augmentation techniques - rephrasing existing questions, performing simple operations (like negation, or change of values) to increase the amount of training data	Shivangi; Shreya
MLM objective : understand the table structure ; "learn from context"	Shreya
Other summarization/encoding methods can be explored-word selection	Shreya
Introducing a way to embed tables rather than selecting random snippets would be helpful.	Seshank
Using logical forms in case of multiple aggregations	Seshank
Make model order invariant-allows tokens from same rows to attend one another and then allows rows to attend one another	Vishal

EXTENSIONS

Visual questions using images	Harman
To make the model work with larger table(in terms of no of rows) we can have a sliding window on the table to generate the embeddings of the whole table and then use layers to select some of those windows for the further aggregation by classification or retrieval.	Aditya
To make it work on composite queries we can add a module that breaks the queries down into a logical combination of simple queries. Then after taking the result of the simpler queries, they can be combined.	Aditya
Handle multilingual QA	Vishal B
New evaluation metrics; complex questions dataset; cases where inductive bias doesn't help	Harman

TABERT: PRETRAINING FOR JOINT UNDERSTANDING OF TEXTUAL AND TABULAR DATA

Pengcheng Yin; Graham Neubig

Carnegie Mellon University

Wen-tau Yih; Sebastian Riedel

Facebook AI Research

TABERT: PRETRAINING FOR JOINT UNDERSTANDING OF TEXTUAL AND TABULAR DATA

- Pretrained LM that jointly learns representations for NL sentences and (semi-) structured tables
- Eg: the task of transducing an NL utterance into a structured query over DB tables
  - (e.g., "Which country has the largest GDP?") into an SQL query
- Understanding structured schema of DB tables
  - (e.g., the name, data type, and stored values of columns)
- Alignment between input text and schema
  - (e.g., the token "GDP" refers to the Gross Domestic Product column)

TABERT: SAMPLE TASK

Show me flights from San Francisco to New York

flight_no	date	leave_from	going_to	state_sold
1	2020-11-14	San Francisco	New York	0
2	2020-11-14	San Diego	San Jose	0
3	2020-11-14	Dallas	Boston	0
4	2020-11-14	Denver	Chicago	0
5	2020-11-14	San Francisco	Sacramento	0
6	2020-11-14	San Luis Obispo	Portland	0

Select flight_no from flights_table where leave_from="San Francisco" and going_to= "New York"

Which country has largest GDP?

#	Country	GDP
1	United States	$19.485 trillion
2	China	$12.238 trillion
3	Japan	$4.872 trillion
4	Germany	$3.693 trillion

Table.argmax(GDP).select(Country)

TABERT: CHALLENGES

Challenges with using LM on tabular data

- Information stored in DB tables exhibit strong underlying structure
- Existing LM are for free form text
- Large number of rows => encoding them is computationally heavy
- Semantic parsing is highly domain specific

TABERT

- Learns contextual representations for utterances and the structured schema of DB tables
- Linearizes table structure to be suitable for Transformer based BERT
- Content snapshots for large tables (Subset of table most relevant to current utterance)
- Vertical attention: Share information/relations across rows

TABERT: CONTENT SNAPSHOT

- Content has more detail than column name
- Works on table content rather than using column names
- Large number of rows; few relevant to current query
- Content snapshot: Row subset relevant to input utterance
- Selecting K rows:
- K>1: highest n-gram overlap ratio with utterance
- K=1: Synthetic row by selecting cell values from each column with highest n gram match.

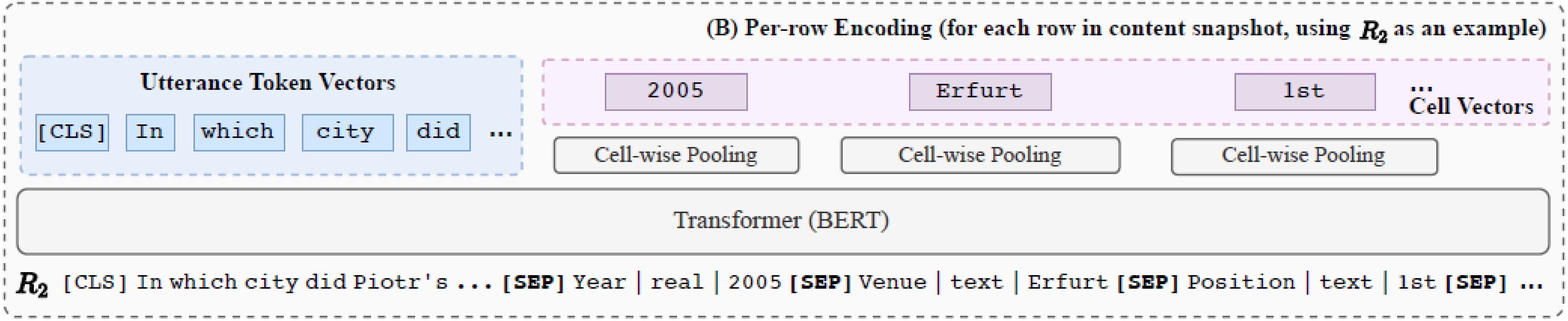
In which city did Piotr's last 1st place finish occur?

	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 : $\{R_2, R_3, R_5\}$

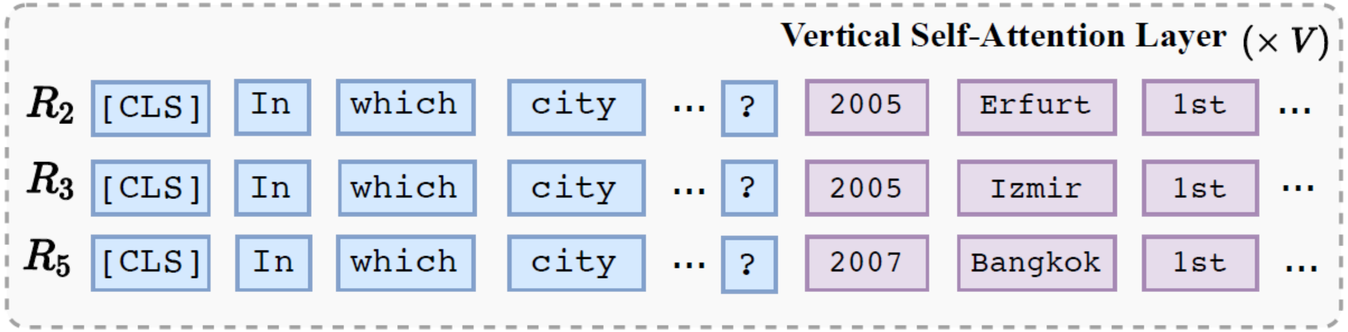
TABERT: ROW LINEARIZATION

- Linearized sequence for rows in the content snapshot as input to Transformer model
- Concatenation for a row (say R2) consists of the utterance, columns, and cell values
- Cell Representation $\underbrace{\texttt{Year}}_{\text{Column Name}} \mid \underbrace{\texttt{real}}_{\text{Column Type}} \mid \underbrace{\texttt{2005}}_{\text{Cell Value}}$
- Cell values separated by [SEP] token

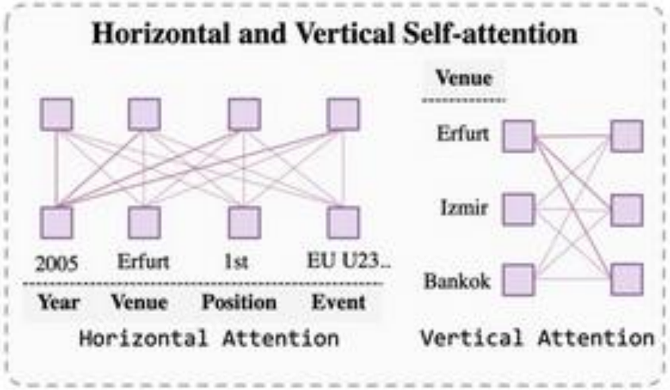


TABERT: VERTICAL SELF ATTENTION

- Allow information flow across cell representations
- Self-attention mechanism over vertically aligned vectors from different rows
- *"V" stacked vertical self attention layers*
- *Aggregate information from different rows*
- *Cross row dependencies captured*

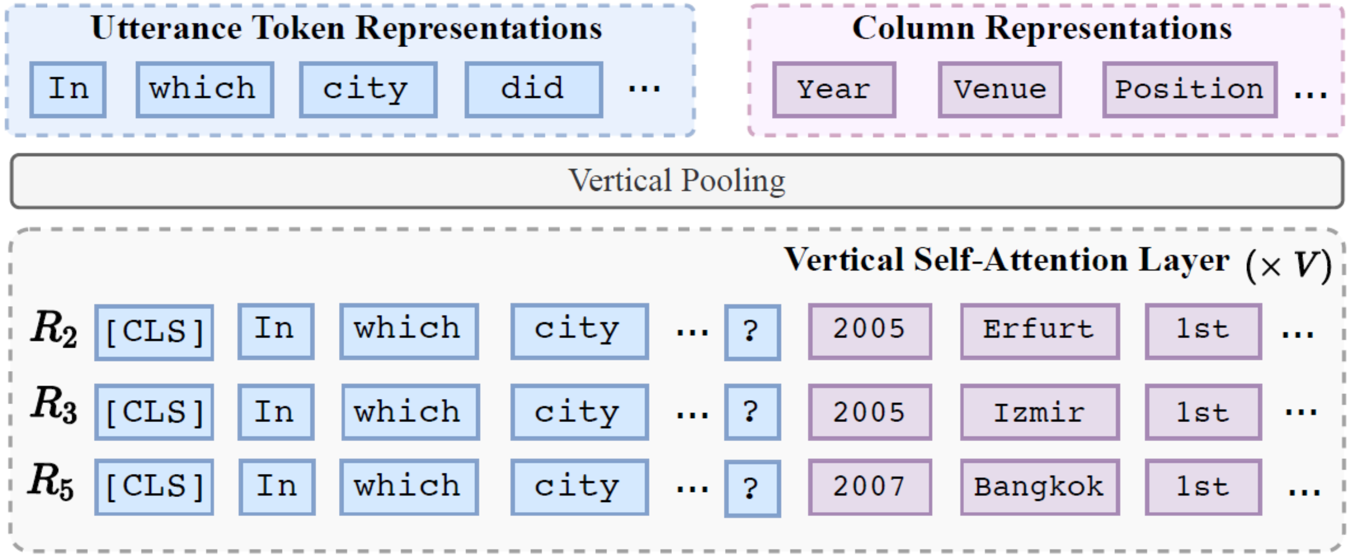


(C) Vertical Self-Attention over Aligned Row Encodings



TABERT: UTTERANCE AND COLUMN REPRESENTATION

- Representation ($\mathbf{c}_i$) of columns c_i
- Mean Pooling over vertically aligned vectors
- *Utterance representation ($\mathbf{x}_j$) computed similarly over vertically aligned tokens*



(C) Vertical Self-Attention over Aligned Row Encodings

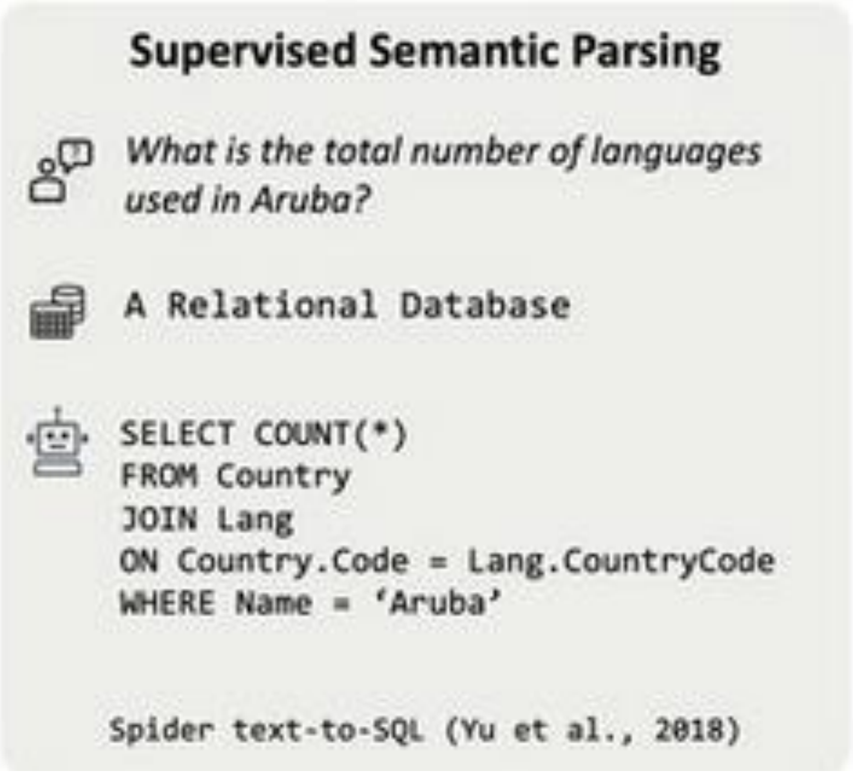
TABERT: PRETRAINING

- Pretraining done on web tables and surrounding text data
- English Wikipedia and the WDC WebTable Corpus, large-scale table collection from CommonCrawl
- 26.6 million parallel examples of tables and NL sentences
- For NL utterances the standard Masked Language Modeling (MLM) objective
- Training column representations:
  - Masked Column Prediction (MCP): mask the column names and datatypes and predict these given column representations c_i
  - Cell Value Recovery (CVR): Predicts the cell tokens given the cell representations $\mathbf{s}_{\langle i,j \rangle}$

TABERT: EXPERIMENTS

- SPIDER Dataset convert text to SQL
- 10,181examples across 200 DBs
- Example consists of
  - NL utterance (What is the total number of languages used in Aruba?")
  - DB with 1 or more tables
  - Annotated SQL query SELECT COUNT(*) FROM Country JOIN Lang ON Country.Code = Lang.CountryCode WHERE Name = `Aruba'

TABERT: EXPERIMENTS



TABERT: RESULTS

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
$w/$ 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
$w/$ 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
$w/$ TABERT$_{\text{Base}}$ (K = 3)	51.6 ±0.5	52.4	51.4 ±0.3	51.3
$w/$ BERT$_{\text{Large}}$ (K = 1)	50.3 ±0.4	50.8	49.6 ±0.5	50.1
$w/$ TABERT$_{\text{Large}}$ (K = 1)	51.6 ±1.1	52.7	51.2 ±0.9	51.5
$w/$ TABERT$_{\text{Large}}$ (K = 3)	**52.2** ±0.7	**53.0**	**51.8** ±0.6	**52.3**

WikiTableQuestions

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

Spider

TABERT: FUTURE WORK

- Try TaBERT on related tasks
- Other table linearization techniques
- Cross lingual settings with tables in English and utterances in other languages

RPT:RELATIONAL PRE-TRAINED TRANSFORMER IS ALMOST ALL YOU NEED TOWARDS DEMOCRATIZING DATA PREPARATION

Nan Tang
QCRI, HBKU

Ju Fan
Renmin University

Fangyi Li et al.
Renmin University

RPT:RELATIONAL PRE-TRAINED TRANSFORMER IS ALMOST ALL YOU NEED TOWARDS DEMOCRATIZING DATA PREPARATION

- Tries to eliminate data preparation, collection and data processing step.
- Pre-trained for tuple-to-tuple model
- Applicable to multiple applications.

RPT:DATA CLEANING

Q1: *r1*[name, expertise, city] = (Michael Jordan, Machine Learning, **[M]**)

A1: Berkeley

Cell Filling

Q2: *r3*[name, affiliation] = (Michael **[M]**, CSAIL MIT)

A2: Cafarella

Value Filling

Q3: *r2*[name, expertise, **[M]**] = (Michael Jordan, Basketball, New York City)

A3: city

Attribute/
Schema Matching

RPT:ENTITY RESOLUTION

	product	company	year	memory	screen
e1	iPhone 10	Apple	2017	64GB	5.8 inchs
e2	iPhone X	Apple Inc	2017	256GB	5.8-inch
e3	iPhone 11	AAPL	2019	128GB	6.1 inches

RPT:INFORMATION EXTRACTION

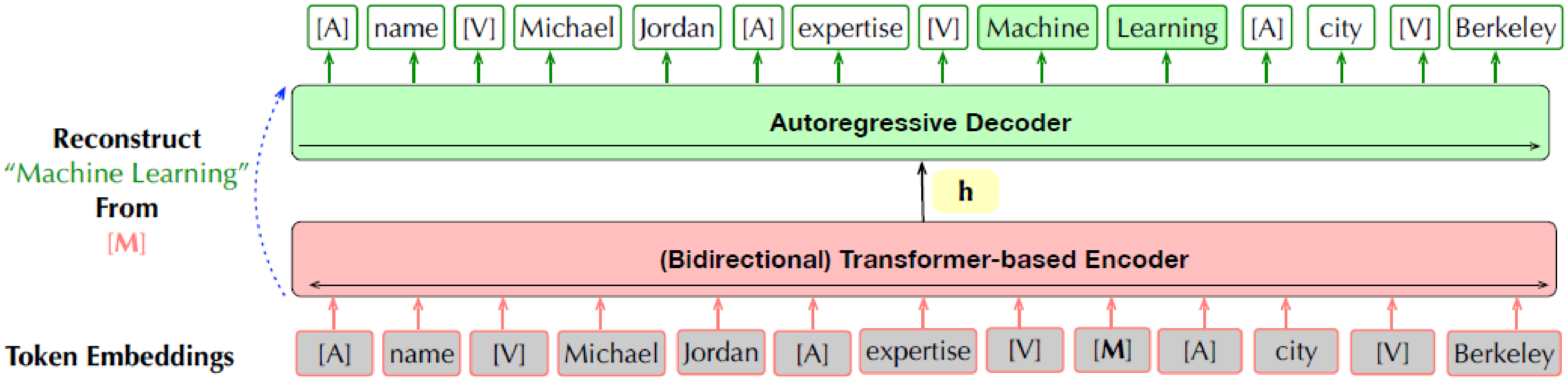
	type	description	label
s1	notebook	2.3GHz 8-Core, 1TB Storage, 8GB memory, 16-inch Retina display	8GB
t1	phone	6.10-inch touchscreen, a resolution of 828x1792 pixels, A14 Bionic processor, and come with 4GB of RAM	4GB

RPT:CHALLENGES

1. Knowledge: Understanding large tables
2. Experience: Learn from other/previous tasks
3. Adaptation: Adjust to new inputs/tasks

RPT:ARCHITECTURE

- Transformer based Bi-dericectional Encoder
- Decoder left to right autoregressive
- Provides flexibility to train on wider range of tasks



RPT: PRE-TRAINING

Tuple tokenization:

- Each tuple as a concatenation of
  - attribute names
  - values

```
name Michael Jordan expertise Machine Learning city Berkeley
```

Token Embedding

- Add Special tokens before attribute[A] and value[V] names

```
[A] name [V] Michael Jordan [A] expertise [V] Machine Learning
[A] city [V] Berkeley
```

Positional and Column Embeddings

- Position and segment embeddings

RPT: PRE-TRAINING

MLM style token masking. Mask tokens with [M] tag.

- Attribute Name Masking: Randomly selected attribute names masked e.g., name
- Entire Attribute Value Masking: Randomly mask full attribute values, e.g., “Machine Learning”
- Single Attribute Value Masking: Randomly mask one of the tokens in attribute value eg:”Machine [M]”

RPT: PRE-TRAINING

Visibility Masking: Restrictive attention learning rules:

- Attribute Name can only attend to
  - Other attribute names
  - Associated tokens
- Token from attribute value can only attend to
  - Attribute values
  - Its own attribute name

RPT: RESULT

Table 1: Compare RPT with BART (yellow: masked values; green: (partially) correct; pink: wrong).

title	manufacturer	price	Truth	RPT-C	BART
instant home design (jewel case)	topics entertainment	[M]	9.99	9	Topics
disney's 1st & 2nd grade bundle ...	disney	[M]	14.99	19	Dis
adobe after effects professional 6.5 ...	[M]	499.99	adobe	adobe	$1.99
stomp inc recover lost data 2005	[M]	39.95	stomp inc	stomp	39.95
[M]	write brothers	269.99	write brothers dramatica ...	write brothers	1.99

RPT: FINE TUNING

- **Value Normalization**: Through sequence generator
  - “Mike Jordan, 9 ST, Berkeley” → “Mike Jordan, 9th Street, Berkeley”
  - Normalizing “Mike” to “Michael” or “Sam” to “Samuel” - neural name translation
- **Data Transformation**: Transformation of data from one format (e.g., a tuple) to another format (e.g., JSON or XML)
- **Data Annotation:** Given a tuple, data annotation requires adding a label (e.g., a classification task)
- **Information Extraction:** Given a tuple, IE extracts a span or multiple spans of relevant text.
- **Entity Resolution**, blocking, entity matching

REFERENCES

REFERENCES

EmbDI

Paper: Riccardo Cappuzzo, Paolo Papotti, and Saravanan Thirumuruganathan. 2020. Creating Embeddings of Heterogeneous Relational Datasets for Data Integration Tasks. In Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data (SIGMOD '20). Association for Computing Machinery, New York, NY, USA, 1335–1349.
DOI:https://doi.org/10.1145/3318464.3389742
Slides:https://dl.acm.org/action/downloadSupplement?doi=10.1145%2F3318464.3389742&file=3318464.3389742.mp4

TAPAS

Paper: Jonathan Herzig and Pawel Krzysztof Nowak and Thomas Müller and Francesco Piccinno and Julian Martin Eisenschlos (2020). TAPAS: Weakly Supervised Table Parsing via Pre-training. *CoRR, abs/2004.02349*.

REFERENCES

TaBERT

Paper: Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for Joint Understanding of Textual and Tabular Data. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8413–8426, Online. Association for Computational Linguistics.
Slides:https://slideslive.com/38929345/tabert-pretraining-for-joint-understanding-of-textual-and-tabular-data

TAPAS

Paper: Jonathan Herzig and Pawel Krzysztof Nowak and Thomas Müller and Francesco Piccinno and Julian Martin Eisenschlos (2020). TAPAS: Weakly Supervised Table Parsing via Pre-training. *CoRR, abs/2004.02349*.

REFERENCES

RPT

Paper: Nan Tang and Ju Fan and Fangyi Li and Jianhong Tu and Xiaoyong Du and Guoliang Li and Sam Madden and Mourad Ouzzani (2020). Relational Pretrained Transformers towards Democratizing Data Preparation [Vision]. *CoRR, abs/2012.02469*.