Representation Learning and Generative Models

for	tabular	data	✨

Madelon Hulsebos (CWI)

TU Berlin

29 January

Agenda of today's lecture

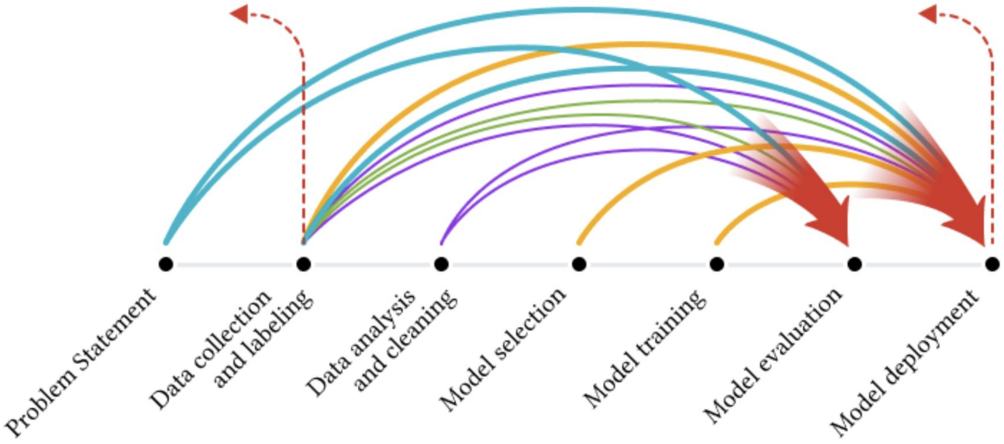
- Why ML for tables?
- Table Representation Learning
  - Background
  - TRL for "data work"
  - TRL for data insights
- Generative models and tabular data
  - Representation learning versus generative models
  - LLMs for (tabular) predictive modeling
  - Agentic data science systems
- Where are we, and where do we go?

Recap

ML pipelines: from raw data to analysis insights of ML model predictions.

80% data work → what happens here, has huge impact!

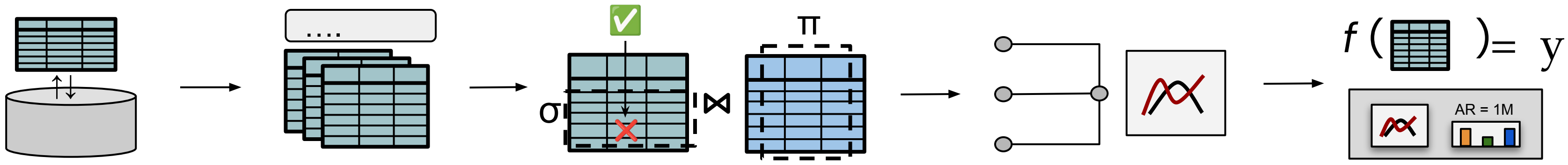
20% model work



From "Everyone wants to do the model work, not the data work": Data Cascades in High-Stakes AI, Sambasivan et al., SIGCHI, 2021

Breaking down a Data Science pipeline

STORE → **RETRIEVE** → **PROCESS** → **ANALYSE** → **USE**



STORE

- indexing
- query execution
- compression

RETRIEVE

- search
- **validation**
- exploration

PROCESS

- transformation
- augmentation
- **cleaning**

ANALYSE

- **exploration**
- statistics
- visualization

USE

- business intelligence
- **predictive modeling**
- ML observability

So much to think about!

So much to go wrong!

Then, I realized: **everyone is doing this**....

What if.... we could use ML to help us do the *data work*, for ML?

→ let’s make data work, model work

→ ML for Data Engineering for ML

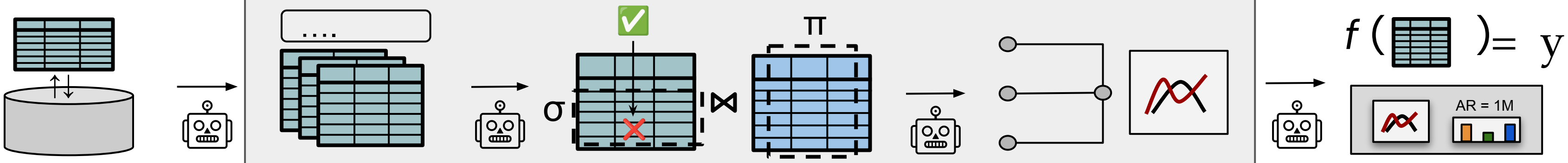
Why would this work?

- “everyone is doing this” = we have data, e.g. CSVs + jupyter notebooks
- If we have data, we can “learn”! ?

Automating data work for ML (predictive modeling)

The ambition...

STORE | **RETRIEVE** | **PROCESS** | **ANALYSE** | **USE**



STORE
- indexing
- query execution
- compression

RETRIEVE
- search
- validation
- exploration

PROCESS
- transformation
- augmentation
- cleaning

ANALYSE
- aggregation
- statistics
- visualization

USE
- business intelligence
- predictive modeling
- ML observability

"data work"

Why "ML for **Tables**"?

From language and vision models → table models

✓

You
what to include in speech with PhD graduation committee after defense ceremor

ChatGPT
After successfully defending your PhD dissertation, it's customary to give a spee
gratitude and appreciation to your graduation committee and others who have su
throughout your academic journey. Here's what you might want to include in your

1. **Acknowledgment of the Committee**: Begin by thanking each member of your gr committee individually, mentioning their contributions to your research and thei during the defense process.
2. **Express Gratitude**: Express your sincere gratitude to your advisor, committee m mentors, colleagues, friends, and family for their support, encouragement, and a throughout your PhD journey.
3. **Reflect on the Journey**: Reflect briefly on your academic journey, highlighting ke challenges you've overcome, and significant moments of growth and learning.
4. **Highlight Contributions**: Summarize the main contributions of your research an impact in your field or related areas. This is an opportunity to showcase the signi work and its relevance.

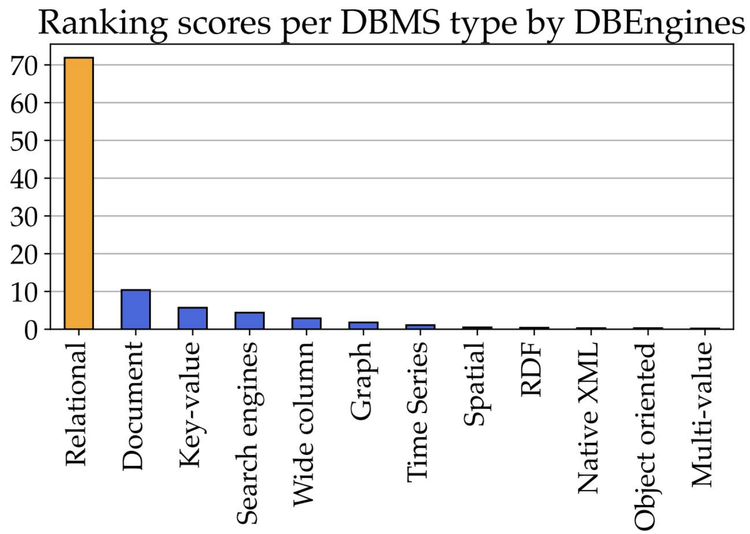
✓

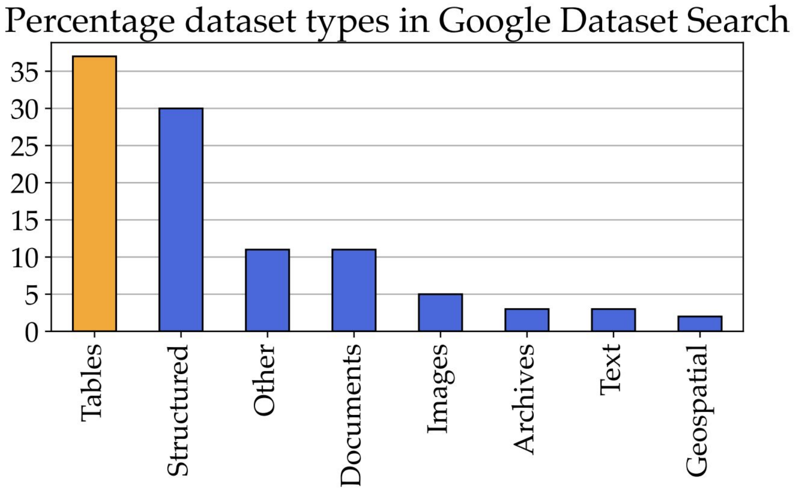
Nr	ID	seed rate	yield	crop	cultivar	pre crop	pre-pre crop	pre-pre-pre
1	68		91	winter wheat		sugar beets	beans	
2	68		100	winter wheat		sugar beets	rotation fallow	
3	68		97	winter wheat		sugar beets	fallow land (5,5y)	
4	136		95	winter wheat		oats	sugar beets	
5	136		96	winter wheat		potatos	sugar beets	
6	136		107	winter wheat		sugar beets	maize	
7	136		107	winter wheat		sugar beetsn	summer wheat	maize
8	136		82	winter wheat		oats	sugar beets	sugar beets
9	136		77	winter wheat		potatos	sugar beets	
10	136		85	winter wheat		sugar beets	maize	maize
11	136		84	winter wheat		sugar beets	summer wheat	sugar beets
12	57	371	98	winter wheat	Sperber	sugar beets	winter barley	winter wheat
13	57	365	98	winter wheat	Sperber	potatos	sugar beets	summer bar
14	57	365	105	winter wheat	Sperber	sugar beets	maize	maize
15	57	365	97	winter wheat	Sperber	sugar beets	winter wheat	sugar beets
16	39	433	90	winter wheat	Okapi	summer barley		
17	39	433	100	winter wheat	Okapi	oats		
18	39	433	97	winter wheat	Okapi	winter wheat		

?

Tables are Everywhere

Data modalities in the real-world data landscape





For a reason: tables serve high-value applications, e.g. data analysis & predictive modeling

A Challenge of Heterogeneity…

Tables store lots of *structured, fresh, domain* data!

Tables come in all shapes, semantics and sizes…

crop rotation : Tabelle

Nr	ID	seed rate	yield	crop	cultivar	pre crop	pre-pre crop	pre-pre-pre	soil type	precipita	tempera	comment
1	68		91	winter wheat		sugar beets	beans		sandy loam, loe	636	9,6	wb, sg,
2	68		100	winter wheat		sugar beets	rotation fallow		sandy loam, loe	636	9,6	cultivation
3	68		97	winter wheat		sugar beets	fallow land (5,5y)		sandy loam, loe	636	9,6	1993-1996
4	136		95	winter wheat		oats	sugar beets		sandy loam, loe	636	9,6	
5	136		96	winter wheat		potatos	sugar beets		sandy loam, loe	636	9,5	cultivation
6	136		107	winter wheat		sugar beets	maize		sandy loam, loe	636	9,5	1991-1994
7	136		107	winter wheat		sugar beetsn	summer wheat	maize	sandy loam, loe	636	9,5	
8	136		82	winter wheat		oats	sugar beets	sugar beets	sandy loam, loe	636	9,5	organic
9	136		77	winter wheat		potatos	sugar beets		sandy loam, loe	636	9,5	organic
10	136		85	winter wheat		sugar beets	maize	maize	sandy loam, loe	636	9,5	organic
11	136		84	winter wheat		sugar beets	summer wheat	sugar beets	sandy loam, loe	636	9,5	organic
12	57	371	98	winter wheat	Sperber	sugar beets	winter barley	winter wheat	sandy loam, loe	635		wb, ww
13	57	365	98	winter wheat	Sperber	potatos	sugar beets	summer barl	sandy loam, loe	635		cultivation, weed
14	57	365	105	winter wheat	Sperber	sugar beets	maize	maize	sandy loam, loe	635		1987-1992
15	57	365	97	winter wheat	Sperber	sugar beets	winter wheat	sugar beets	sandy loam, loe	635		
16	39	433	90	winter wheat	Okapi	summer barley			sandy loam, loe	690	8,5	oats, cultivation, weec
17	39	433	100	winter wheat	Okapi	oats			clay, silt	690	8,5	1982-1986
18	39	433	97	winter wheat	Okapi	winter wheat			clay, silt	690	8,5	

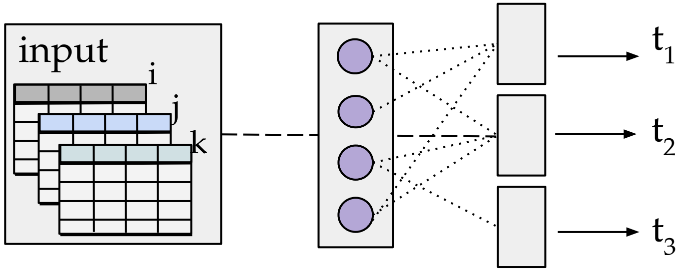
	2019			delta		
	Profit	Quantity	Sales	Quantity	Sales	Profit
Overall	128.9k	13.3k	1.0m	36.5%	36.2%	30.9%
France	35.1k	3.9k	308.4k	33.8%	33.2%	8.7%
Austria	7.5k	299.0	24.6k	13.3%	11.9%	40.8
Belgium	4.2k	202.0	17.3k	23.2%	82.7%	98.6%
Denmark	-1.3k	86.0	2.8k	4.9%	-21.1%	22.9%
Zealand	-245.0	15.0	242.7			
South Denmark	-362.9	40.0	1.3k	33.3%	-7.7%	41.1
Sonderborg	-45.4	6.0	87.5			
Odense	-280.3	30.0	1.1k	0.0%	-20.1%	54.5
Esbjerg	-37.3	4.0	88.7			
Hovedstaden	-0.7k	31.0	1.3k	-40.4%	-40.9%	35.4%
Copenhagen	-0.7k	31.0	1.3k	0.0%	-29.5%	23.
Frederiksberg	232.1	11.0	1.1k			
Finland	36.0k	2.4k	216.5k	-79.6%	-81.4%	-86.2%
Germany	-3.9k	152.0	7.2k	15.0%	46.3%	32.9%
Ireland	10.6k	1.4k	109.7k	17.8%	101.6%	-152.5%
Italy	-11.6k	0.6k	23.7k	37.5%	37.2%	57.6%
Netherlands	2.9k	135.0	12.9k	46.3%	12.6%	2.5%
Norway	-1.0k	74.0	1.8k	229.3%	230.8%	162.8%
Portugal	20.6k	1.2k	99.1k	-29.5%	-79.2%	82.4%
Spain	-9.4k	360.0	15.6k	46.7%	12.9%	27.1%
Sweden	2.2k	84.0	7.3k	143.2%	167.7%	-128.7%
Switzerland	36.8k	2.4k	194.0k	5.0%	-25.6%	-29.1%
United Kingdom	1.3k	48.0	4.0k	68.4%	6.4%	32.7%
Wales	1.9k	118.0	6.6k	-5.9%	-30.2%	-28.7%
Scotland	33.6k	2.2k	183.4k	0.9%	-54.6	-55.5

Challenging… how to deal with variation?

Representation Learning for tabular data

Table Representation Learning

Map each **table** to some consistent input. **Learn** some **representation** that helps detect patterns relevant to given task(s).



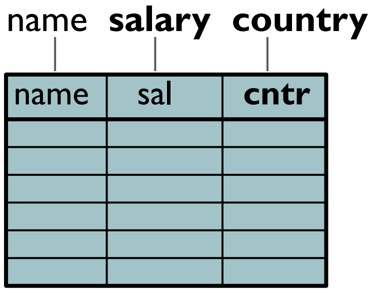
TRL for data work

Table Semantics Are All You Need

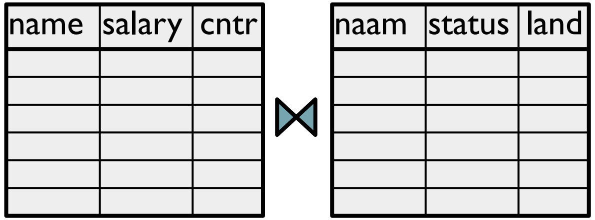
A table's understanding comes through its columns.

ML task:

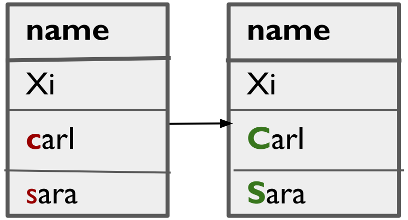
- given table *T*,
- predict semantic column types C,
- with each c in C from preset ontology.



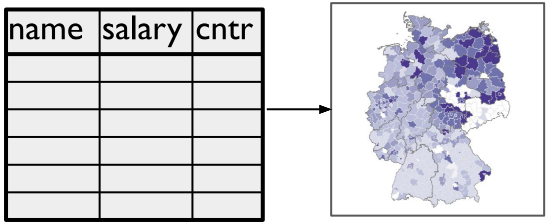
Semantic column types dictate operations *sensible* to perform on them:



Inform semantic join on tables



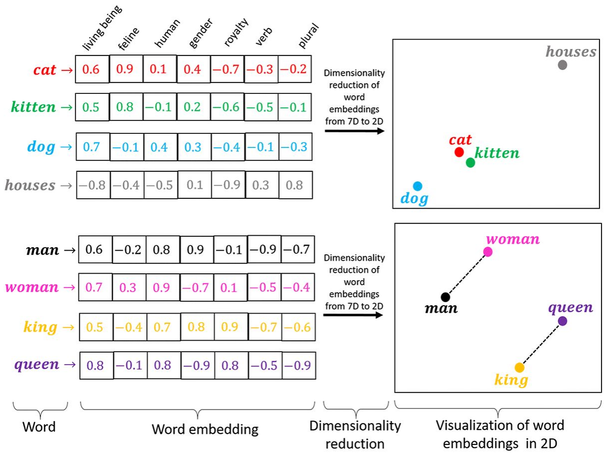
Capitalize “name” columns



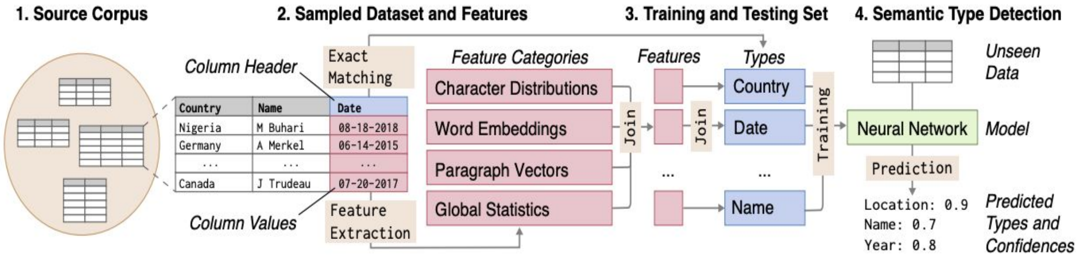
Plot “country” data

Semantic column types

Word embeddings: represent "words" in numeric vector space reflecting semantics



Starting point: treat a column as set of strings



Hulsebos et al., Sherlock: A deep learning approach to semantic data type detection. KDD, 2019.

Fast Forward to Transformers

- Bottleneck of existing Deep Learning models:
  - Don’t take in much context (1 input column -> 1 output label)
  - Not very scalable

- Transformer architecture: **attention** mechanism enables “contextual learning” in parallel!

Attention Is All You Need

Ashish Vaswani* Google Brain avaswani@google.com

Noam Shazeer* Google Brain noam@google.com

Niki Parmar* Google Research nikip@google.com

Jakob Uszkoreit* Google Research usz@google.com

Llion Jones* Google Research llion@google.com

Aidan N. Gomez* † University of Toronto aidan@cs.toronto.edu

Łukasz Kaiser* Google Brain lukaszkaiser@google.com

Illia Polosukhin* ‡ illia.polosukhin@gmail.com

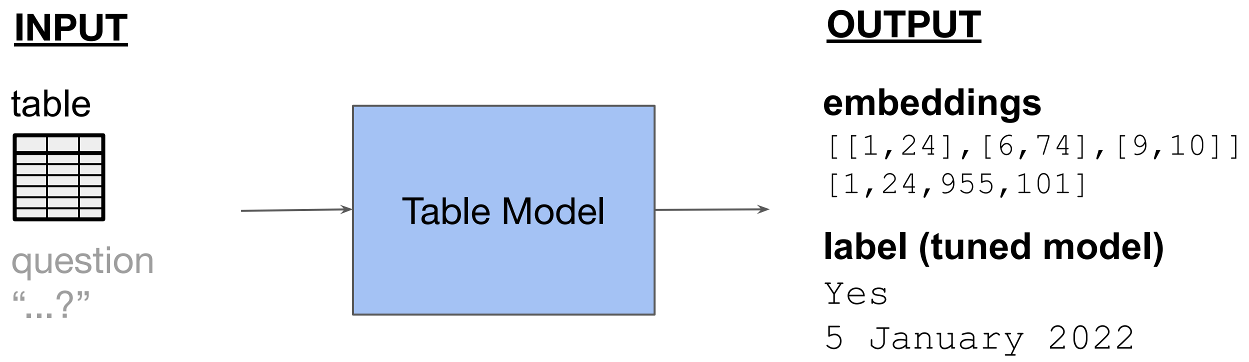
Start of LMs.. read!

Transformers for Tables

High level pipeline

More context, more efficient -> lower level training!

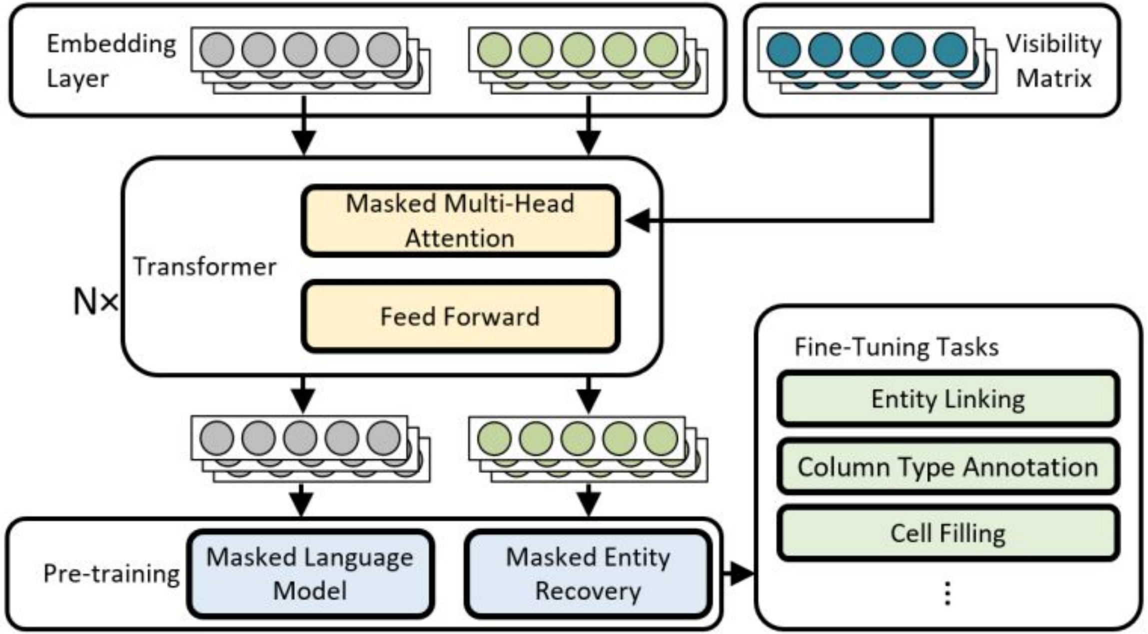
Example task: question answering over tables



Based on: Models and Practice of Neural Table Representations. Hulsebos, Deng, Sun, & Papotti. SIGMOD 2023.

“Table Model”: TURL

Low level inputs and architecture

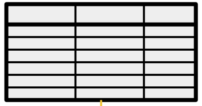


Deng et al., Table Understanding through Representation Learning, 2023, VLDB

Transformers for Tables

Input: from Table to Tokens

- Sample, serialize, and tokenize(+pad) table.
- Table can be aligned w/ metadata or other input (if any).
- Many variations for serialization (e.g. row-wise, w/ SEP tokens etc).



“abc dz”?

`[ abc dz? ]`

`[ col : col1|…|colN row1 : val1|val2|…|valN row2 : … ]`

SERIALIZE

`[token1, token2, token3, …, tokenM]`

TOKENIZE

`[1, 2, 3, 4, 5, 6, …, M]`

TOKEN → TOKEN ID

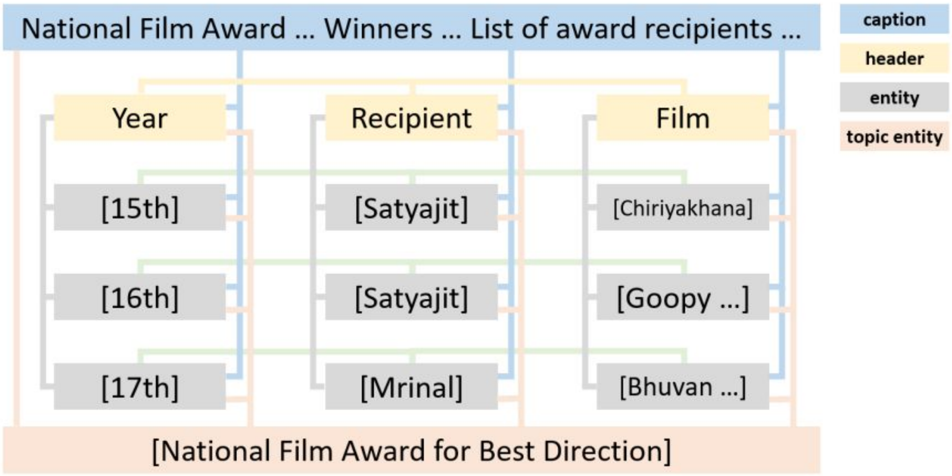
Token ID -> token embedding -> embedding is learned

Transformers for Tables

Architecture: Learning Adjusted for Table Structure

- Attention learns across all tokens (context) in input text.
- But `Mrinal` has not much to do with `Goopy` → structural attention:

→ Structural attention: vertical = across column or matrix = across row/col (TURL).

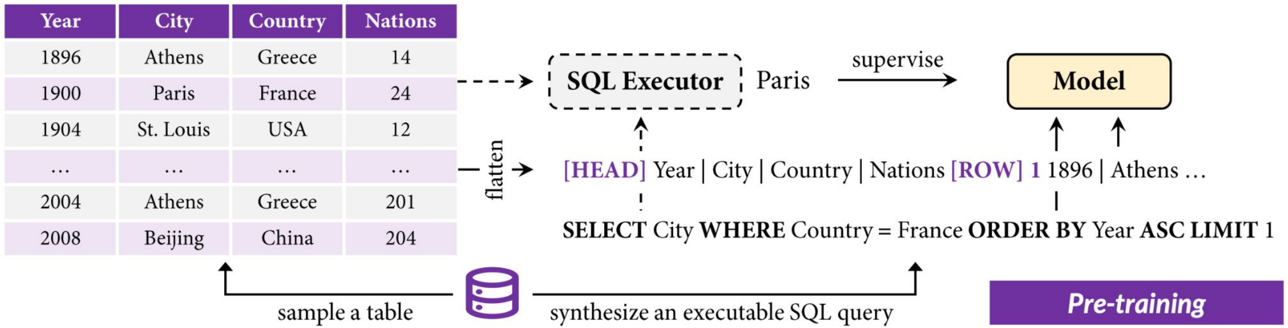


Transformers for Tables

Pre-training: Table-specific Tasks

Pretraining because the goal is to obtain “generic model” that can be “fine-tuned” for various tasks (using “embeddings” of inputs)

- Pre-training tasks:
  - Typical: recovering (predicting) column names or cell values.
  - Efficient: (synthesized) SQL execution (↓ TaPEx [1]).

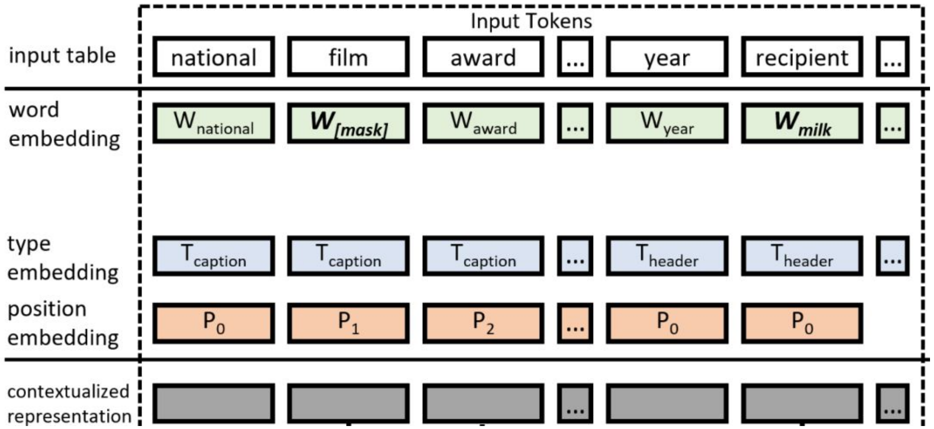


*Liu et al, TAPEX: Table pre-training via learning a neural SQL executor, ICLR, 2022.

Transformers for Tables

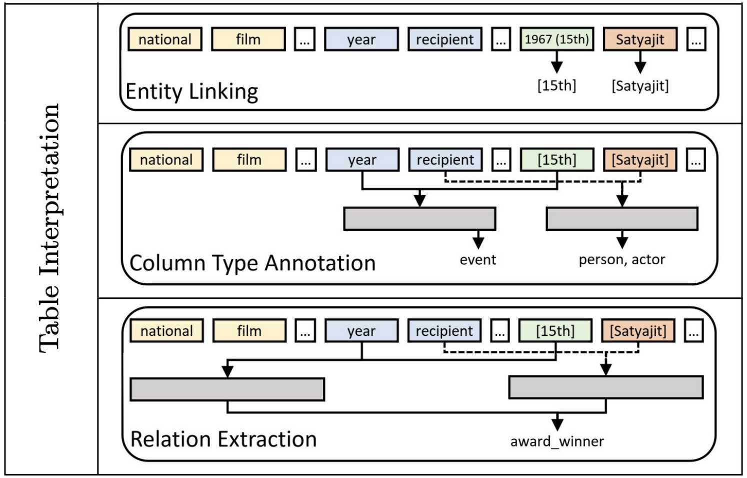
Output: Embeddings or Predictions

Embeddings



Typically aggregated from token-level embeddings to cell/row/col level

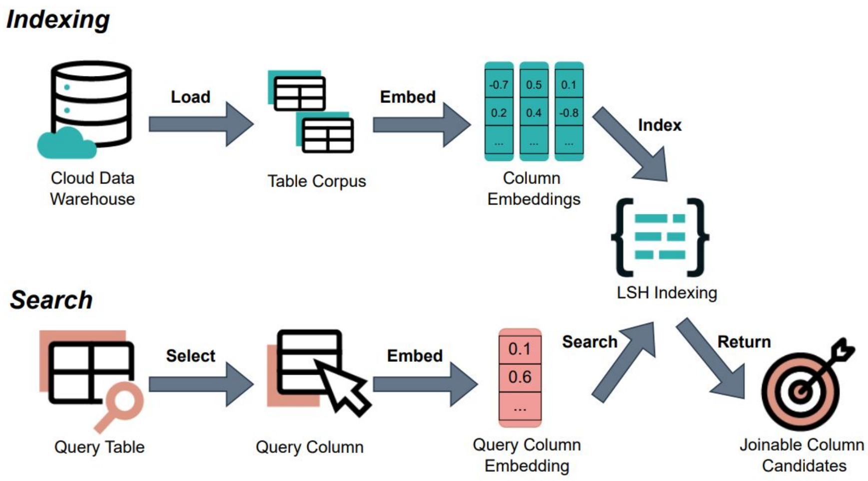
Predictions



These are "fine-tuned" from embeddings to explicit labels

Representation Learning for Join Discovery

Task: given input table, find joinable tables for given column.



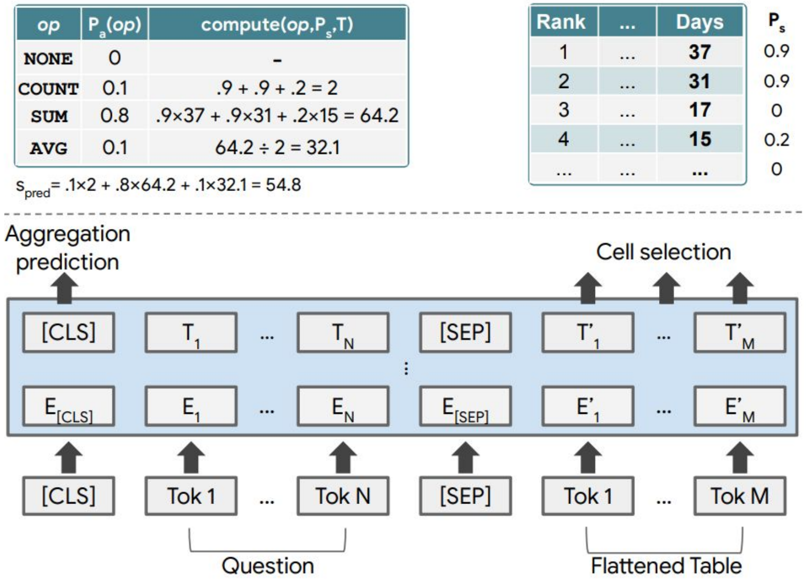
But embeddings used for retrieval, correlation prediction, data validation, etc.

Cong et al., WarpGate: A Semantic Join Discovery System for Cloud Data Warehouses, CIDR, 2023

TRL for data insights

TRL for Question Answering over Tables

TaPas: TRL model for QA predicting operator + cell span



Herzig et al, TAPAS: Weakly Supervised Table Parsing via Pre-training, ACL, 2020.

Works Well… for Basic Cases

Example with TaPas:

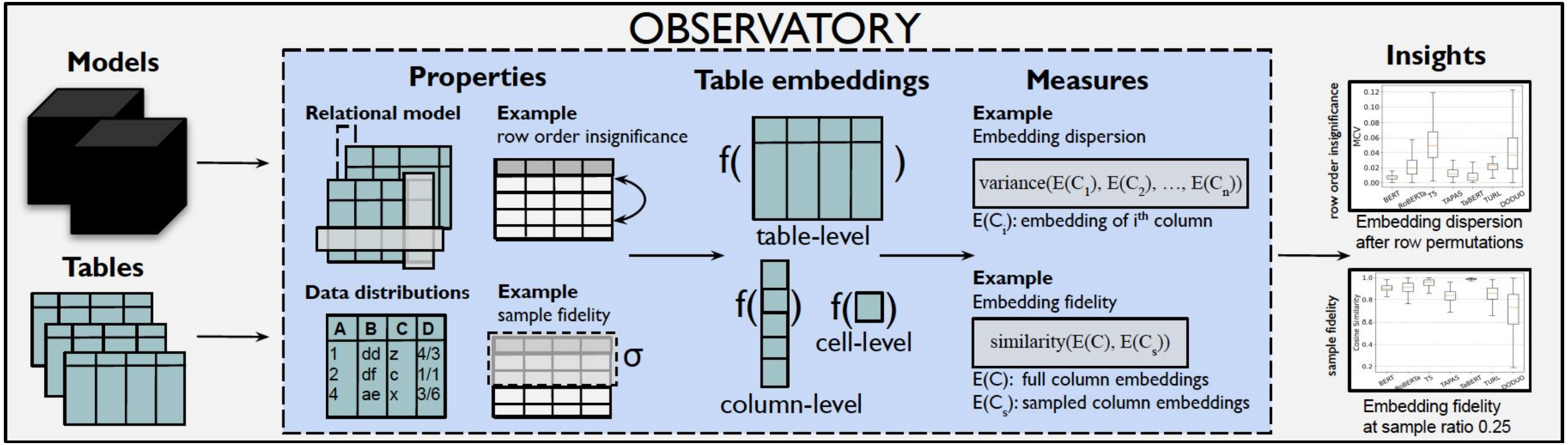
Actor	Age	Number of movies
Brad Pitt	56	87
Leonardo Di Caprio	45	53
George Clooney	59	69

How old is Leonardo Di Caprio?	AVERAGE	45
What is Leonardo Di Caprio his age?	AVERAGE	45
What is the sum of the number of movies?	SUM	87, 53, 69 = 209
How many movies are there in total?	COUNT	87, 53, 69 = 3

Do Table Embeddings capture Relational Properties?

Tables ≠ natural language ?

Studying neural table embeddings through **Codd's relational model.**



Observatory: Characterizing Relational Table Embeddings. Cong, <u>Hulsebos</u>, Sun, Groth, Jagadish, VLDB, 2024.

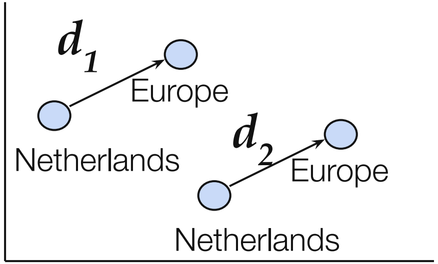
Example Property: Functional Dependencies

Given table with FD: X=country → Y=continent

ID	name	country	continent
1	Kathryn	Netherlands	Europe
2	Oscar	Netherlands	Europe
3	Lee	Canada	North America
4	Roxanne	USA	North America
5	Fern	Netherlands	Europe
6	Raphael	USA	North America
7	Rob	USA	North America
8	Ismail	Canada	North America

We argue that:

- FD relations interpretable as *translation* between embeddings E($\pi X\,(s)$) and E($\pi Y\,(s)$)
- Model preserves FD if d(E($\pi X(s)$), E($\pi Y(s)$)) = d(E($\pi X(t)$), E($\pi Y(t)$)) where *d* preserves magnitude+direction (L1/L2-norm).
- Intuitively:



Current Architectures Fall Short…

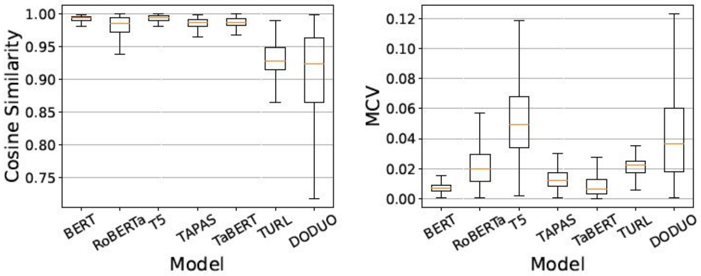
Turns out, most models do not preserve FDs!

We also consider simpler properties:

A *relation* then consists of a set of tuples, each tuple having the same set of attributes. If the domains are all simple, such a relation has a tabular representation with the following properties.

(1) There is no duplication of rows (tuples).
(2) Row order is insignificant.
(3) Column (attribute) order is insignificant.
(4) All table entries are atomic values.

Measure by avg cosine similarity of col embeddings across row permutations.



row order robustness

Impact downstream tasks: **row shuffling affects 34%** semantic column types!

Generative Models for tabular data

Representation Learning vs Generative Models

From **predicting** labels, e.g.:

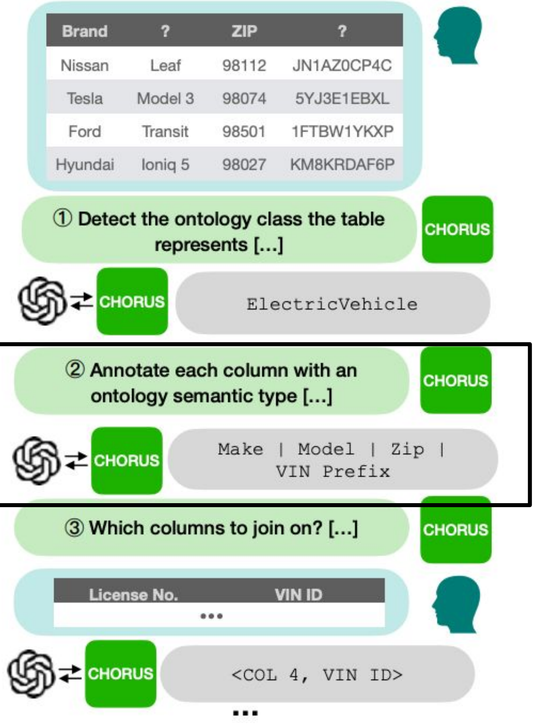
- semantic types / relations between columns,
- cell span + aggregations (QA),
- *True* or *False* (fact verification).

To **generating** answers…!

- Input: table, context, query (question / task / anything)
- Output: anything (e.g. code, or explicit answers)
- *Underlying* mechanism: **next-token *prediction***

“Underlying” because “underlying” LM might be “tuned” to predict discrete labels.

Can LLMs help with data discovery?




```
Consider this example. Input:
```
Name, Famous Book, Rk, Year
Fyodor Dostoevsky, Crime and Punishment, 22.5, 1866
Mark Twain, Adventures of Huckleberry Finn, 53, 1884
Albert Camus, The Stranger, -23, 1942
```
Output:
`dbo:author, dbo:title, Unknown, dbo:releaseDate`.
For the following CSV sample, suggest a DBPedia.org Property for each
column from the `dbo:`namespace.
```[...]```
```

**(b) Column-type annotation**

Typical format:

- Instructions
- Example (input, output)

Kayali et al., “Chorus: Foundation Models for Unified Data Discovery and Exploration”
```

Transformers Not Always SOTA

Problem: pretrained TRL models poor OOD performance on *col type prediction*.

Just use generative model? Sherlock outperforms LLMs.

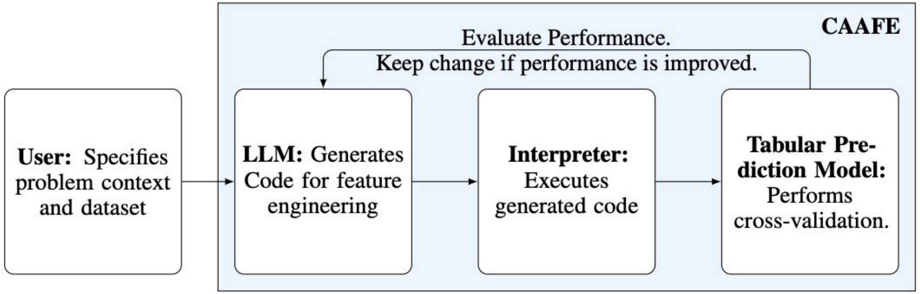
	F_1-score	Precision	Recall
DoDuo-VizNet*	0.900	90.3%	89.9%
Sherlock*	**0.930**	**92.2%**	**93.1%**
TaBERT	0.380	38.9%	38.3%
DoDuo-Wiki	0.815	82.6%	81.4%
Chorus	**0.865**	**90.1%**	**86.7%**

LLM analyses show tables best formatted w HTML tags, but many challenges.

Messy data? Large tables? Full DBs? Non-descriptive headers? Numeric data?

Beyond SQL: feature engineering!

General approach:



Example, binning:

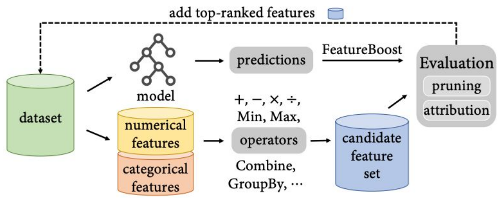
```
# Feature: AgeGroup (categorizes passengers into age groups)
# Usefulness: Different age groups might have different likelihoods
    of being transported.
# Input samples: 'Age': [30.0, 0.0, 37.0]
bins = [0, 12, 18, 35, 60, 100]
labels = ['Child', 'Teen', 'YoungAdult', 'Adult', 'Senior']
df['AgeGroup'] = pd.cut(df['Age'], bins=bins, labels=labels)
df['AgeGroup'] = df['AgeGroup'].astype('category')
```

Automated features helpful, but minimal gain

No feature engineering at all...

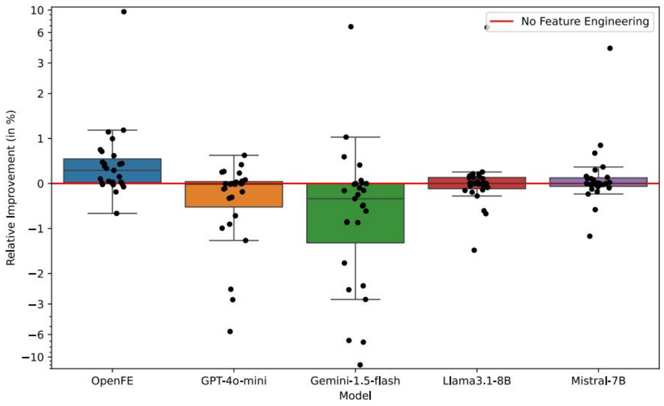
		No FE	Baselines				CAAFE	
			DFS	AutoFeat	FETCH	OpenFE	GPT-3.5	GPT-4
Log. Reg.	Mean	0.749	0.764	0.754	0.76	0.757	0.763	**0.769**
	Mean Rank	27.4	**23.6**	26.2	25.2	25	24.8	24.3
Random Forest	Mean	0.782	0.783	0.783	0.785	0.785	0.79	**0.803**
	Mean Rank	23.4	22.1	21.8	23.5	22.3	23.1	**19.9**
ASKL2	Mean	0.807	0.801	0.808	0.807	0.806	0.815	**0.818**
	Mean Rank	12.2	12.9	12.6	13.4	13.5	**10.9**	11.6
Autogluon	Mean	0.796	0.799	0.797	0.787	0.798	0.803	**0.812**
	Mean Rank	17.6	15.4	16.4	17.6	16.6	15.8	**14.1**
TabPFN	Mean	0.798	0.791	0.796	0.796	0.798	0.806	**0.822**
	Mean Rank	13.9	15	14.8	16.5	13.9	12.9	**9.78**

Generating feature pool,
prune feature candidates



What's going on?

Language models engineer too many simple features…



Better with domain expertise? Or much better training (data, tricks)….

Küken et al., Large Language Models Engineer Too Many Simple Features for Tabular Data, TRL workshop @ NeurIPS, 2024

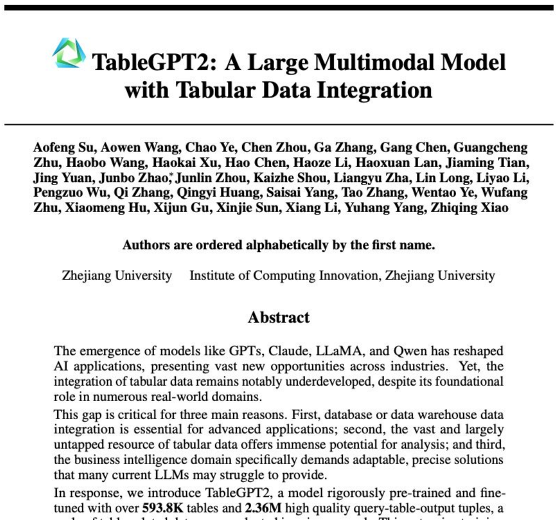
Make LMs significantly better for tabular tasks?

Train LLMs on tabular tasks, **at scale**:

- GPT-based model trained on 86B tokens
- >593.8K table+language samples for training encoder
- >2.36M query+table+output tuples for fine-tuning

Scale of evaluation:

- **23 benchmarking metrics**
- TableGPT2 **7B** model: +35.20% improvement
- TableGPT2 **72B** model: +49.32% improvement



TableGPT2: A Large Multimodal Model with Tabular Data Integration

Aofeng Su, Aowen Wang, Chao Ye, Chen Zhou, Ga Zhang, Gang Chen, Guangcheng Zhu, Haobo Wang, Haokai Xu, Hao Chen, Haoze Li, Haoxuan Lan, Jiaming Tian, Jing Yuan, Junbo Zhao*, Junlin Zhou, Kaizhe Shou, Liangyu Zha, Lin Long, Liyao Li, Pengzuo Wu, Qi Zhang, Qingyi Huang, Saisai Yang, Tao Zhang, Wentao Ye, Wufang Zhu, Xiaomeng Hu, Xijun Gu, Xinjie Sun, Xiang Li, Yuhang Yang, Zhiqing Xiao

Authors are ordered alphabetically by the first name.

Zhejiang University Institute of Computing Innovation, Zhejiang University

Abstract

The emergence of models like GPTs, Claude, LLaMA, and Qwen has reshaped AI applications, presenting vast new opportunities across industries. Yet, the integration of tabular data remains notably underdeveloped, despite its foundational role in numerous real-world domains.

This gap is critical for three main reasons. First, database or data warehouse data integration is essential for advanced applications; second, the vast and largely untapped resource of tabular data offers immense potential for analysis; and third, the business intelligence domain specifically demands adaptable, precise solutions that many current LLMs may struggle to provide.

In response, we introduce TableGPT2, a model rigorously pre-trained and fine-tuned with over **593.8K** tables and **2.36M** high quality query-table-output tuples, a

Impression of scale

Benchmark	Metric	GPT-4o	TableLLM (Qwen2)	TableLLM (CodeQwen)	TableLLM (LLaMA3)	TableLLM (LLaMA3.1)	TableLLM (DeepSeek)	TableLLM-13B	DeepSeek-lite	Yi-Coder	Qwen2.5-Coder	Qwen2.5-Instruct	TableGPT2-7B	TableGPT2-72B
Table Understanding														
Col Type Annot.	F1	31.75	10.10	5.71	1.47	1.59	6.04	12.70	20.58	5.38	32.59	22.19	**85.88**	85.67
Relation Extract.	F1	52.95	1.60	3.79	2.39	2.00	3.34	18.16	8.67	2.25	31.00	15.92	**83.35**	79.50
Entity Linking	Acc	90.80	47.10	39.70	0.20	0.60	15.50	66.25	70.15	41.75	71.70	82.25	92.00	**93.30**
Row Pop.	MAP	53.40	2.20	5.14	1.93	6.23	3.13	14.25	1.20	1.00	13.23	12.30	**59.97**	55.83
Question Answering														
HiTab	Exec Acc	48.40	11.74	0.00	0.00	0.00	39.08	6.30	0.76	0.00	1.70	10.73	70.27	**75.57**
FetaQA	BLEU	21.70	12.24	8.69	2.42	3.10	7.94	10.83	15.08	11.17	13.00	16.91	28.97	**32.25**
HybridQA	Acc	58.60	27.12	20.14	27.35	27.61	19.53	51.88	42.58	29.83	51.10	51.13	53.17	**56.41**
WikiSQL	Acc	47.60	46.50	37.20	39.26	39.00	36.14	41.10	38.30	25.34	46.90	47.42	53.74	**57.32**
WikiTQ	Acc	68.40	64.16	36.05	34.95	38.84	36.05	66.30	47.65	43.37	**74.50**	68.55	61.42	71.45
Fact Verification														
TabFact	Acc	74.40	72.00	53.20	40.06	27.13	60.76	68.95	62.27	79.6	77.26	84.60	77.80	**85.43**
FEVEROUS	Acc	71.60	20.10	46.90	51.50	42.30	18.39	21.45	7.80	38.10	60.70	63.30	**78.05**	76.80
Table to Text														
ToTTo	BLEU	12.21	6.95	3.10	5.50	6.23	3.81	5.36	8.76	2.64	10.50	11.91	14.10	**22.69**
Natural Language to SQL														
BIRD(dev)	Exec Acc	-	9.13	7.37	1.83	2.48	0.39	0.72	25.10	24.19	27.18	18.97	31.42	**38.40**
BIRD(dev-knowledge)	Exec Acc	-	15.45	18.19	3.39	3.72	0.39	1.83	36.51	39.96	42.96	31.42	49.28	**60.76**
Spider(dev)	Exec Acc	-	42.26	32.88	12.86	18.96	2.71	4.26	66.44	58.12	70.99	61.70	76.31	**79.40**
Spider(test)	Exec Acc	-	40.29	34.93	12.02	16.35	7.33	2.93	66.65	56.87	69.73	60.18	74.38	**78.48**
Holistic Table Evaluation														
TableBench	DP	-	26.62	26.44	26.71	26.73	26.15	3.88	29.60	21.94	28.67	25.18	32.03	**38.90**
	TCoT	-	37.08	31.33	29.79	30.01	28.65	3.85	30.93	22.8	36.25	29.77	42.34	**50.06**
	SCoT	-	14.11	17.78	9.60	12.38	22.39	2.88	22.61	8.43	25.95	24.35	25.01	**30.47**
	PoT@1	-	21.05	26.39	31.96	25.80	28.39	2.94	10.90	11.36	16.15	22.58	**33.52**	28.98

LLMs for predictive modeling

Problem setting

f(**X**) → **y**

Where **X** are *features*, and **y** is the *target* to predict.

Quite similar to missing value imputation, where **y_test** are missing values?

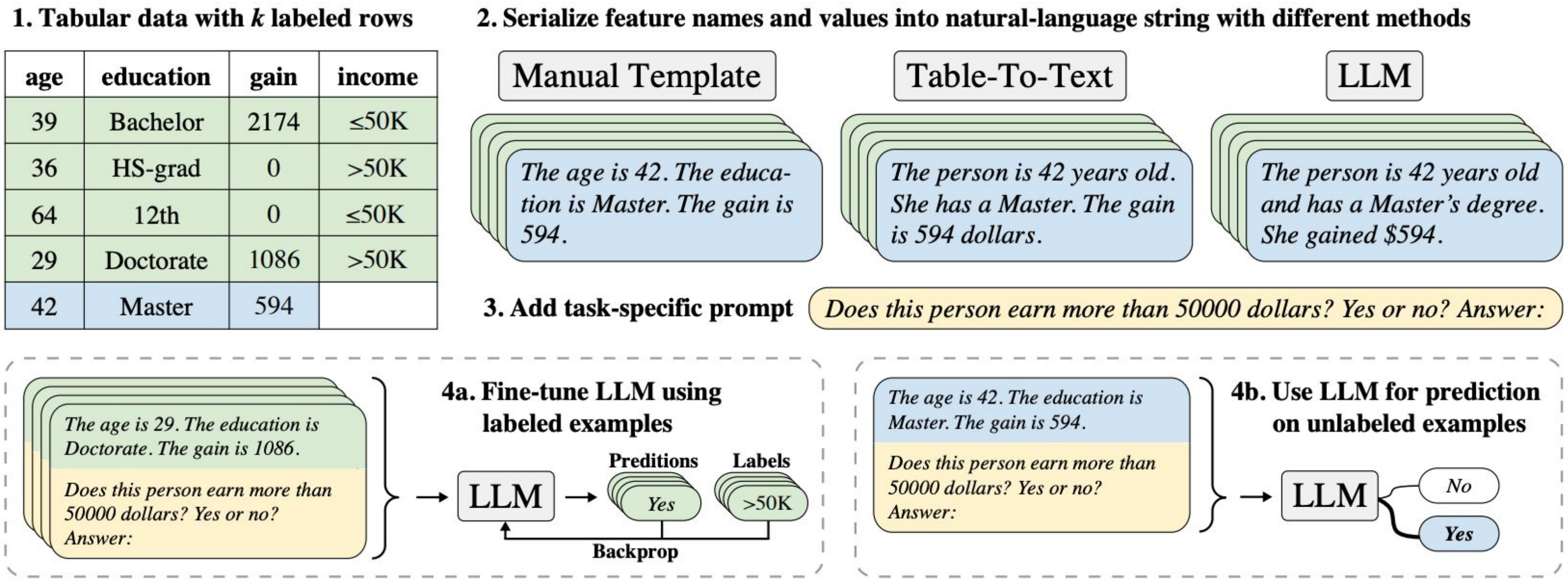
General approach

Given **generative model M** and a **table T**:

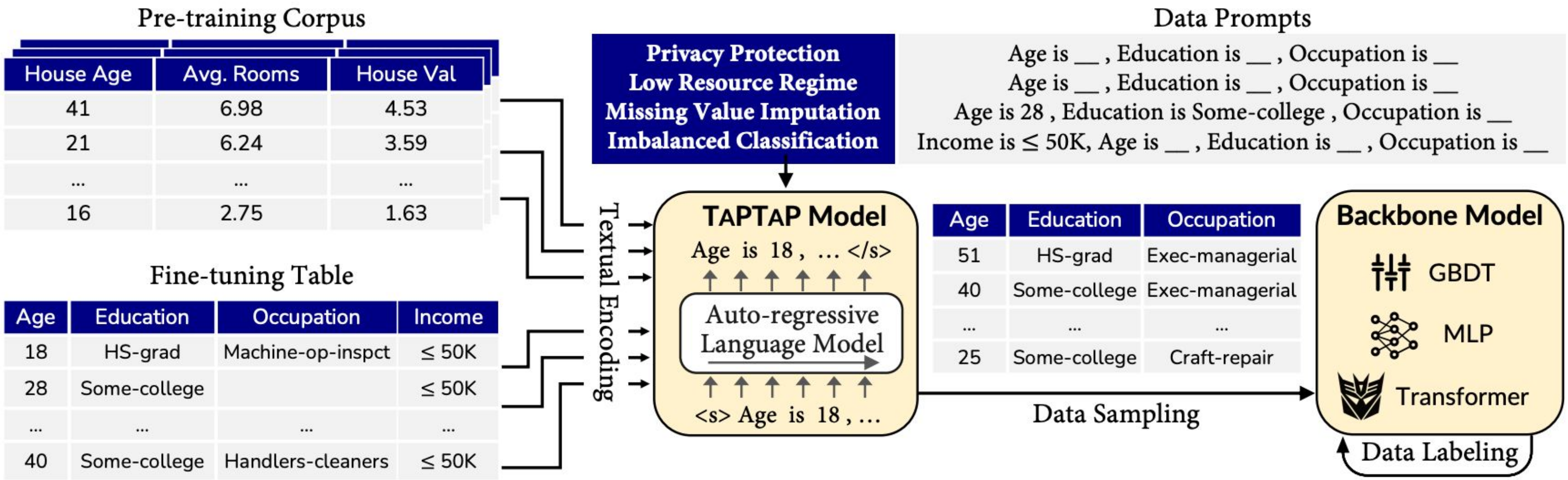
- Serialize rows in **T** into “sentences”
- Template the prediction “task”
- Fine-tune **M** on train set (where to-be-predicted labels are provided)
- Evaluate on test set (labels to-be-predicted)

→ Still “generation”, so prone to errors in hallucination, formatting, etc.

TableLLM: few-shot LLMs for predictive modeling



TapTap: Language Models for Predictive Models

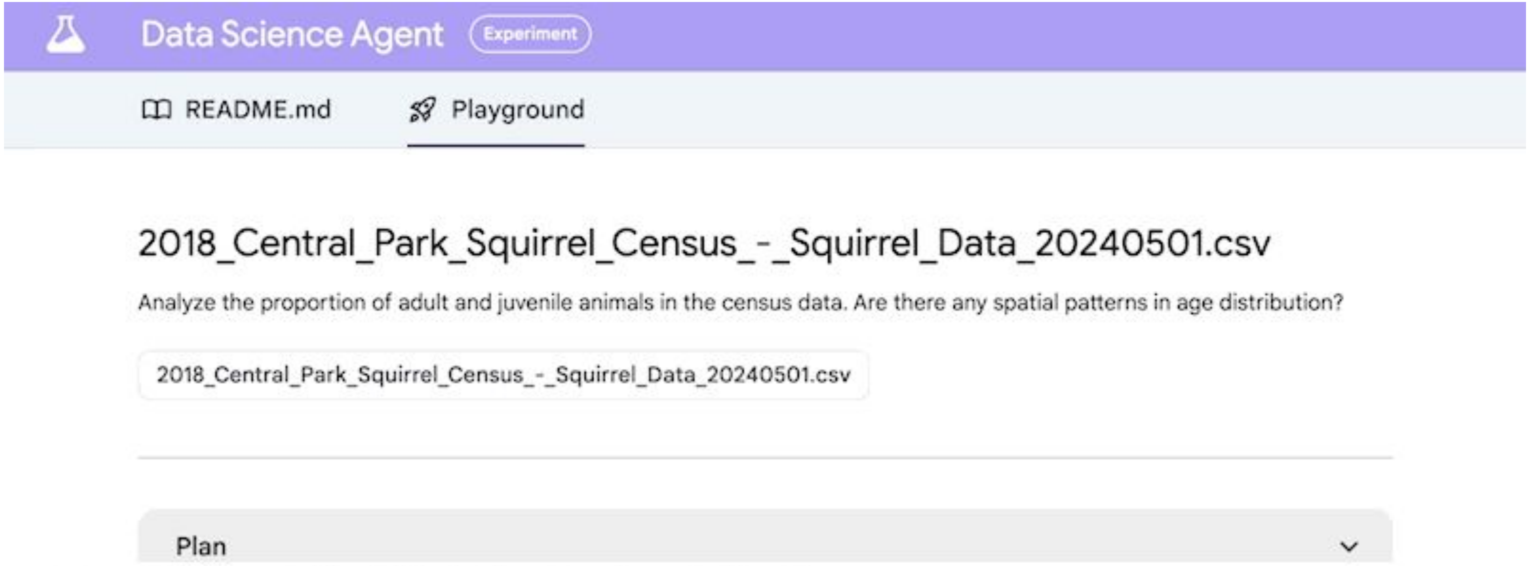


Note: data augmentation with TapTap, improving robustness to invariant row/col order!

Agentic Data Science

Agentic Systems for Data Science

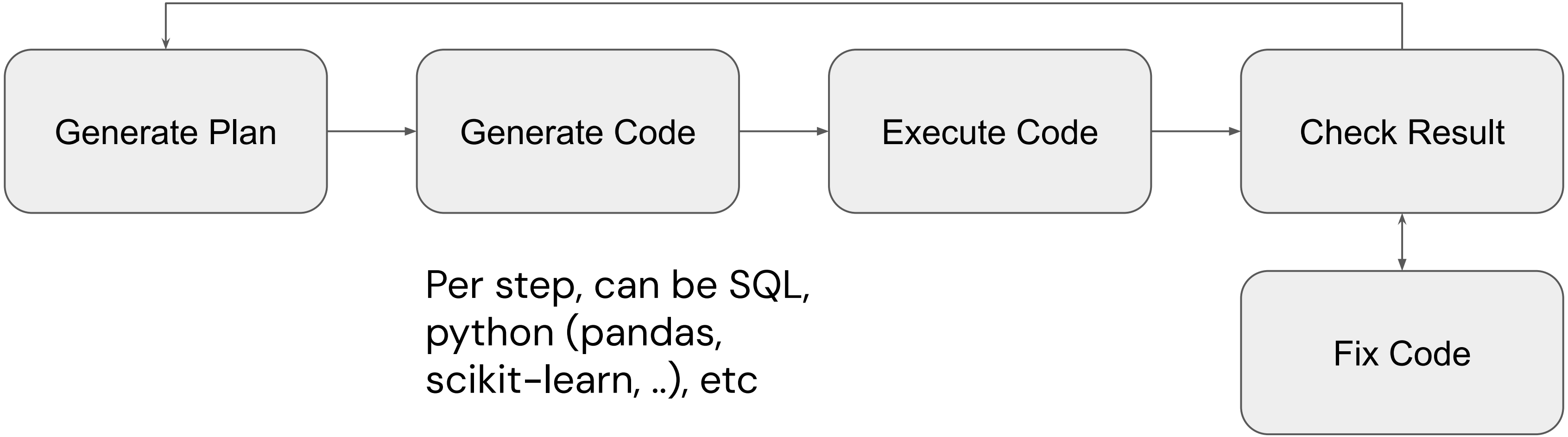
“Agentic”: the LLM-system has some “agency”, i.e. it plans what to do.



Pipeline of 8 steps, <u>*automated*</u>! (8, but didn’t even train/eval an ML model)

One step, e.g. “data cleaning”

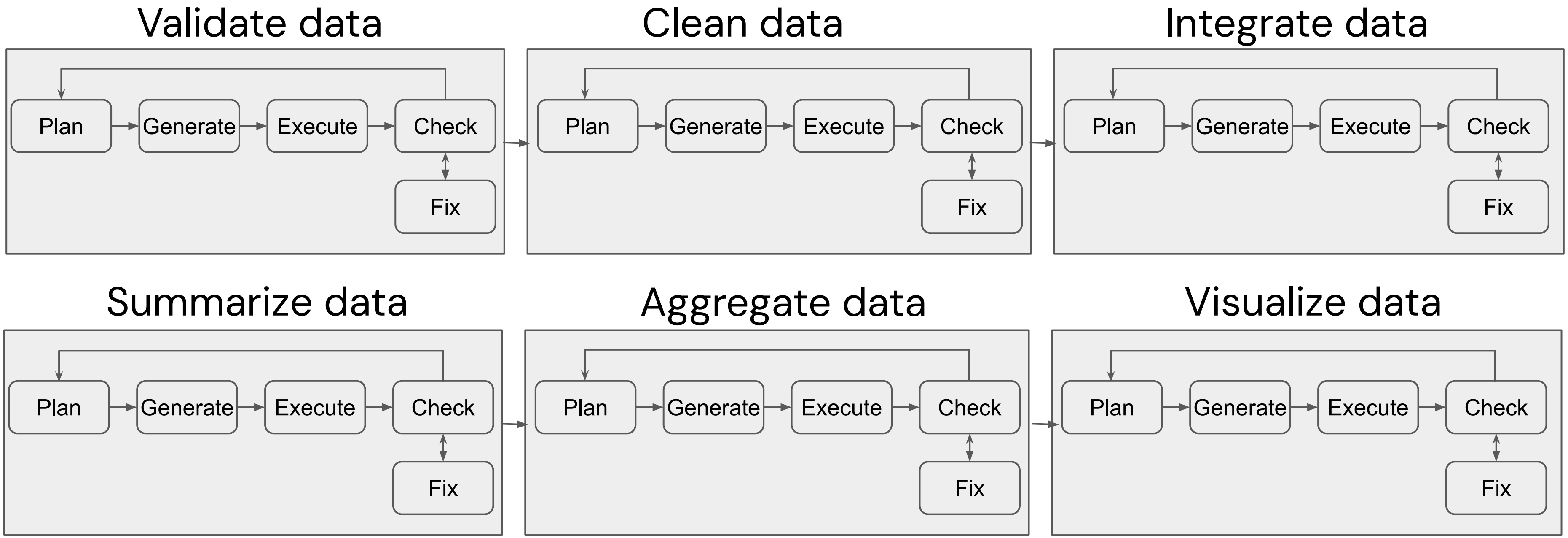
Data cleaning:
- Remove invalid values → Nice, an LLM can reason about what “invalid” would mean, examples?
- Remove outliers
- Impute missing values
- …



Per step, can be SQL, python (pandas, scikit-learn, ..), etc

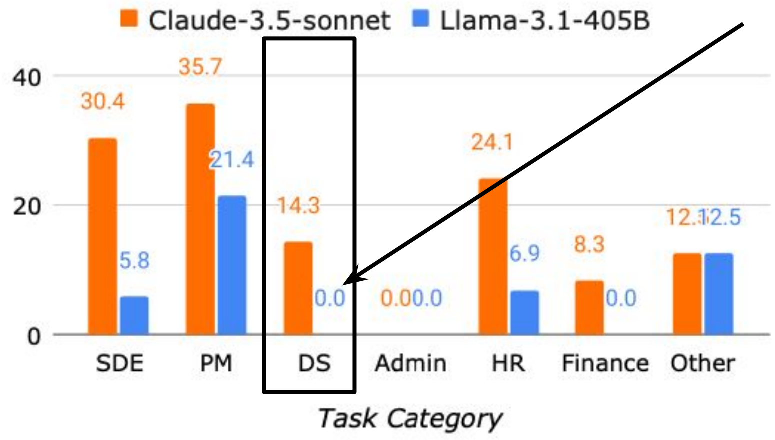
What can possibly go wrong?

Analyze the proportion of adult and juvenile animals in the census data.
Are there any spatial patterns in age distribution?



How do agentic DS systems perform?

Realistic data science (DS) task



(b) Success rate across task categories

Model	*SDE* (69 tasks) Success	*SDE* (69 tasks) Score	*PM* (28 tasks) Success	*PM* (28 tasks) Score	*DS* (14 tasks) Success	*DS* (14 tasks) Score
Closed model APIs						
Claude-3.5-Sonnet	30.43	38.02	35.71	51.31	14.29	21.70
Gemini-2.0-Flash	13.04	18.99	17.86	31.71	0.00	6.49
GPT-4o	13.04	19.18	17.86	32.27	0.00	4.70
Gemini-1.5-Pro	4.35	5.64	3.57	13.19	0.00	4.82
Amazon-Nova-Pro-v1	2.90	6.07	3.57	12.54	0.00	3.27
Open-weight models						
Llama-3.1-405b	5.80	11.33	21.43	35.62	0.00	5.42
Llama-3.3-70b	11.59	16.49	7.14	19.83	0.00	4.70
Qwen-2.5-72b	7.25	11.99	10.71	22.90	0.00	5.42
Llama-3.1-70b	1.45	4.77	3.57	15.16	0.00	5.42
Qwen-2-72b	2.90	3.68	0.00	7.44	0.00	4.70

Xu, Song, Li, et al., “TheAgentCompany: Benchmarking LLM Agents on Consequential Real World Tasks”, 2024.

Some suggestions…

- Errors are costly!
  - Need for human interaction – “how” is an open question (reviewing code 🤔?)
  - Better interpretation (refinement) of input query.
- Generalizability is key
  - Robustness to variation (data, workflow needs)
  - Need to acknowledge limitations (current demo mode: “can do, will do!”)

But… promising!

Key take-aways

- Potential of TRL & generative models for tables for data work!
- LLMs can do predictive modeling, reasonably
- We're moving towards agentic Data Science systems

More attention needed to:

- We need representative and large-scale datasets (hard to get!)
- We need specialized "tricks" (e.g. architecture, pretraining, tokenization, etc)
- We need better domain context and ways to fetch human guidance

Got interested?

Join us for a workshop on this topic on 27 February in Amsterdam:

ELLIS workshop on Representation Learning and Generative Models for Structured Data

ellis | UNIT AMSTERDAM

https://sites.google.com/view/rl-and-gm-for-sd/home
Or check: https://www.madelonhulsebos.com/**upcoming/**

Thank you!

Madelon Hulsebos
TRL Lab @ CWI

https://www.madelonhulsebos.com
@madelonhulsebos on Bluesky