Fine-Tuning LLMs for SQL Execution Tasks on Proprietary Data Using QATCH

Moaad MAAROUFI

EURECOM

June 26, 2024

Introduction

- Table Representation Learning (TRL) focuses on extracting meaningful features and patterns from tabular data.
- Models are pre-trained on large open-domain datasets to support downstream tasks (e.g NLP-based fact-checking systems/Data Imputation/Table Question).
- Companies tend to adopt a pre-trained model to reduce costs and fine tune it on their proprietary data. Papicchio et al. QATCH: Benchmarking SQL-centric tasks with Table Representation Learning Models on Your Data,2023
- Fine-tuning models on proprietary data isn't straightforward; good performance on open benchmarks doesn't guarantee similar results on proprietary data.Papicchio et al.

QATCH (Query- Aided TRL Checklist)

- QATCH Papicchio et al. is a toolbox providing reliable performance metrics for models and automatically generating natural language questions and equivalent SQL query templates based on user-proprietary tables for **Question Answering** (QA, where the model returns an answer given a natural language question and a table) and **Semantic Parsing**(SP, where the model returns a SQL query given a natural language question and a ta- ble schema) tasks.
- Examples of templates for queries in SQL and natural language from the paper:

Category	SQL declaration	Free-Text question
Project	SELECT $\{c_1, \ldots, c_n\}$ FROM $\{T\}$	Show $\{c_1, \ldots, c_n\}$ in table $\{T\}$
Distinct	SELECT DISTINCT $\{c_1, \ldots, c_n\}$ FROM $\{T\}$	Show the different $\{c_1, \ldots, c_n\}$ in table $\{T\}$
Select	SELECT * FROM $\{T\}$ WHERE $\{c_i\}$ { op }{ val }	Show data of table { t } where $\{c_i\}$ { op }{ val }
Order by	SELECT * FROM $\{T\}$ ORDER BY $\{c_i\}$ { ord }	Show data for table $\{T\}$ in { ord } order by $\{c_i\}$

QATCH Cross-task performance metrics

Performance metrics provided by QATCH to evaluate the models on QA and SP:

- **Cell Precision** :The fraction of table cells in the output instances that are relevant to the input query.
- **Cell Recall** The fraction of table cells that are relevant to the input query that are successfully retrieved.
- **Tuple constraint** The fraction of ground truth tuples in the query output. Most important metric
- **Tuple cardinality** The ratio of output and ground truth cardinality.

Problem Formulation

We try to use QATCH in this study to answer the following questions:

- Does fine-tuning with synthetic QATCH examples improve Table Question Answering (TAPAS, transformer architecture)? (Herzig et al., Tapas: Weakly supervised table parsing via pre-training, 2020)
- Can we get comparable results across different models (TAPEX)?(Liu et al., Tapex: Table pre-training via learning a neural sql executor, 2022)
- Does fine-tuning domain-specific/proprietary data break the model's (TAPAS) performance on the original generic dataset (Spider)?Yu et al., Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task, 2019
- Training and testing models on simple/template-based QATCH data don't reflect true performance. How will models perform if trained on complex/realistic LLM variations of QATCH domain-specific questions?

Why are these questions relevant?

- In (Li et al., Table-gpt: Table-tuned gpt for diverse table tasks,2023b), even after fine-tuning, ChatGPT and Table-ChatGPT still perform the worst in the QA answering category among all the others (Error Detection, Data Imputation, Row-to-Row Transformation...), even after the fine tuning the performance bump is minimal.
- Fine-tuning and maintaining small specialized models is cost-effective and ecologically friendly compared to behemoth models like ChatGPT.
- Company workloads may change over time. They may still want the model to perform well on both old and new workloads, so fine-tuning on new workloads shouldn't reduce (significantly) performance on the old ones.

Datasets: The Spider Dataset and The domain-specific/proprietary data

- The SPIDER dataset (named so because it is cross-domain and complex)
- **10,181** questions and **5,693** unique complex SQL queries on **200** databases with multiple tables covering **138** different domains
- As proprietary data, we will use some domain-specific tables from Kaggle (four categories), diverse in terms of data properties (different sizes and arity). These are the same ones used in the paper Papicchio et al.. We will also check if we can boost performance with the fine-tuning (no fine-tuning in the paper).

Kaggle domain-specific tables

Figure from the paper Papicchio et al.

Category	Table Name	# rows	# categorical cols	# numerical cols	Example cols
ECOMMERCE	Sales-transactions	500k	5	3	ProductNo, Date
	Fitness-trackers	565	8	3	Brand Name, Display
FINANCE	Account-fraud	1M	4	26	DaysSinceRequest, Velocity6h
	Late-payment	2466	6	6	InvoiceDate, Disputed
MEDICINE	Heart-attack	303	1	11	# trtbps, # oldpeak
	Breast-cancer	686	5	6	pgr, rsstime
MISC	Adult-census	32.6k	9	6	education, fnlwgt
	Mushrooms	8.1k	23	0	cap-shape, ring-type

Data Preprocessing: Spider

- Neither QATCH nor the used models support some query types, which we need to filter out.
- Queries that touch on more that one table, or tables with more than 15 rows. (TAPAS input is limited to 512 tokens table+question, TAPEX to 1024)
- Remove the ones using these operations: 'join', 'union', 'intersect', 'except', 'inner', 'outer', 'left', 'right', 'full', 'limit', 'like'.
- Check for nested queries by counting the occurrences of 'select' and remove them.
- Ensure there is at most one aggregation keyword: 'count', 'sum','avg', 'min', 'max'.
- Either aggregation or projection not both at the same time: SELECT C1, MIN(C1) FROM table GROUP BY C1
- Instead we should only have : SELECT MIN(C1) FROM table GROUP BY C1

Data Preprocessing: Spider

- Need to fetch the tables from SQLite databases and put them in dataframes for the models.
- The Hugging Face training pipeline(tokenzier) for both models requires the answers to the queries (answer coordinates for TAPAS only) to calculate the loss (The Cell Selection Loss and Scalar Answer Loss in case of aggregation).
- Answers are easy to fetch, but the coordinates(top left most cell in the pd dataframe is (0,0)) are problematic.(no order in the SQLite databases).
- Use script provided by TAPAS and modify it to to take the first occurrence in case of duplicate answers. For aggregation only accept single value answer.
- Spider has a training (**7000** rows) set and Dev set (**3000** rows) which we use for testing. The filtering brings them down to **1400** and **200** respectively. The trainset is split into 80/20 for training and validation.

Data Generation and prepossessing: QATCH

- Generate data given SPIDER trainset tables: we get 21,000 questions and their SQL query equivalents; we do not use all of them.
- Generate data for the proprietary Kaggle tables after truncating them to 15 rows each; 2 tables per domain give approximately 400 rows in synthetic examples.
- Note that QATCH-generated examples are skewed towards the HAVING clause; we do not control the number of generated examples.
- These synthetic examples go through the same prepossessing steps as mentioned before for uniformity.

Data Generation and prepossessing: QATCH question variations using LLMs

- QATCH uses simple template-based generation for the natural language questions $\Longrightarrow$ increased risk of overfitting+ testing is not realistic (data leakage).
- Use a large language model (LLM) with few-shot learning to create different variations of the same question, using synonyms and different wording.
- Experiment with OpenAI's GPT-3.5(cheap), and finalize with GPT-4o(cheaper and faster than GPT-4)
- Some questions arise:
  1. how "complex" should the questions be? In other words, should the questions assume that the user knows the exact schema, as Spider and QATCH-generated questions do? How much should we assume that the user knows? $\Longrightarrow$ generate both: queries in the same style as Spider and queries which suppose that the user can only describe the schema in a human way and doesn't know the exact names.

Data Generation and prepossessing: QATCH question variations using LLMs

- should we consider the temperature as a hyperparameter, or is prompt engineering enough? $\Longrightarrow$ no need for another hyper parameter, use default (0) temperature so that the LLM follows the instructions closely.
- What should we provide to the LLM: the NL question or the SQL? $\Longrightarrow$ Provide only the sql queries otherwise the LLMs tend to get influenced by the QATCH style of the question, which uses the exact schema and often refuses to actually use synonyms and different wording.

Data Generation and prepossessing: QATCH question variations using LLMs

- **The most important question**: how we can we ensure that the mapping between the LLM-generated natural language (NL) question and the original SQL query is maintained? $\Longrightarrow$ Use OpenAI's **embedding model text-embedding-3-large model** (ranked 19th on the Massive Text Embedding Benchmark (MTEB) Leaderboard (Muennighoff et al., 2023)), to measure the cosine similarity between the generated NL question and the original one.
- The results are overoptimistic and not very reliable.
- Two questions with similar lengths and use the exact schema names, the score is inflated even if the operations are not the same.
- Conversely, if the LLM-generated question is longer, the score is low even though the mapping is conserved.

Data Generation and prepossessing: QATCH question variations using LLMs

- Abandon the idea of verifying the mapping as after checking the variations, they are quite good most of the time.
- The questions that don't conserve the mapping will act as regularizers.
- If the variations are bad we will see it in the fine-tuning as the model performance will drop.
- Examples of question variations , using constrained generation without relinquishing all the control to LLM but guiding it through the generation process. **112** QATCH queries $\Longrightarrow$ **560** LLM variations, **5** for each query

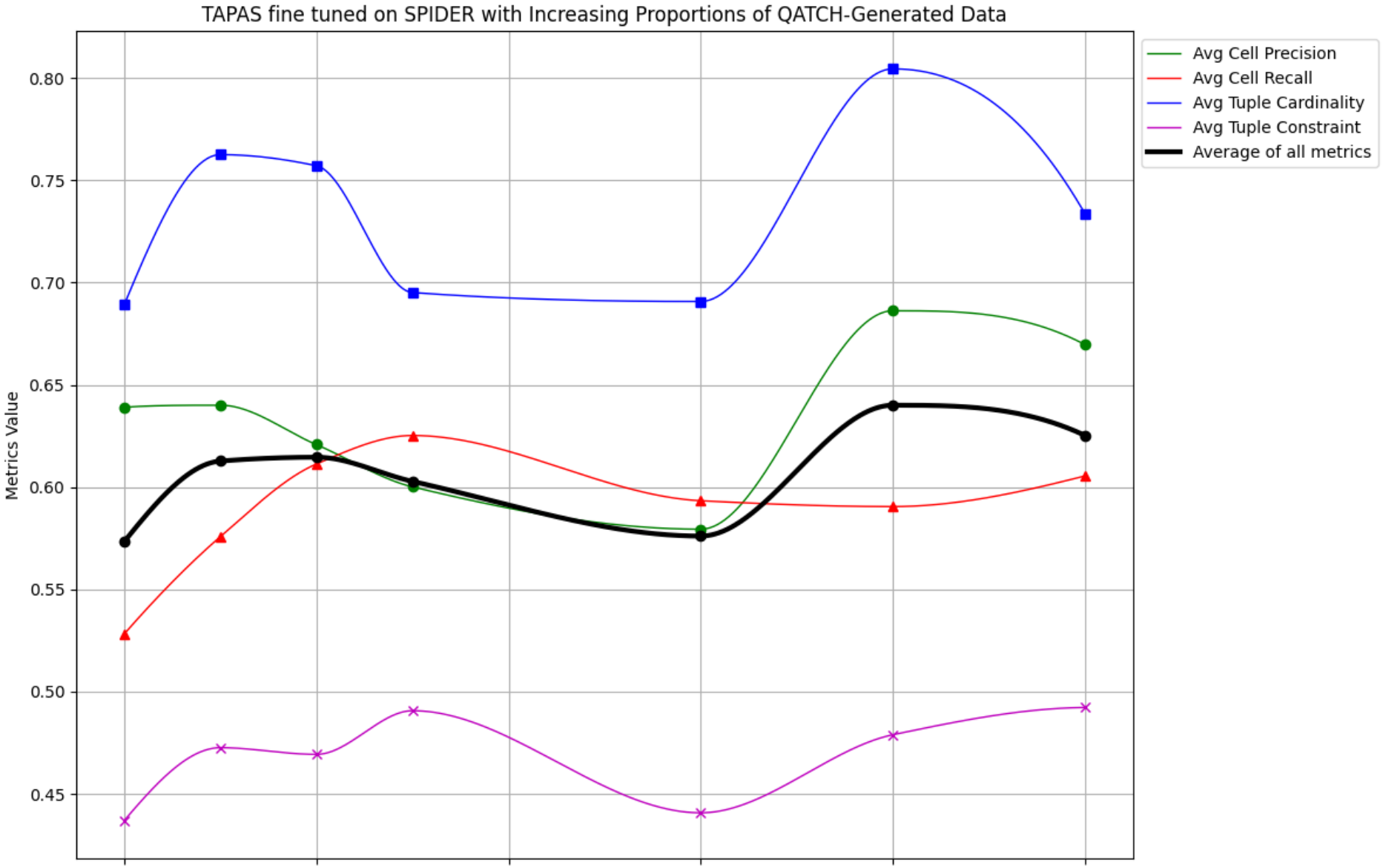
Data Generation and prepossessing: QATCH question variations using LLMs

- SELECT DISTINCT restingelectrocardiographicrresults, output, chestpaintype, fastingbloodsugar, sex FROM heartAttack $\Longrightarrow$ Can you list the unique combinations of ECG results, outcomes, chest pain types, blood sugar levels, and gender from the heart attack data?
- SELECT hormonaltherapy FROM breastCancer ORDER BY hormonaltherapy DESC $\Longrightarrow$ Can you list the types of hormonal treatments sorted from the most to the least in the breast cancer data?
- SELECT sex FROM heartAttack GROUP BY sex HAVING AVG(numberofmajorvvessels) $\leq 0.77 \Longrightarrow$ From the heart attack records, which gender shows an average of major vessels being 0.77 or less?

Experiments: Spider +QATCH sythtic data

- The first experiment: A baseline to prove that models can benefit from QATCH data to boost performance on a generic dataset:
  1. Train the model on the Spider dataset (0% QATCH data).
  2. Incrementally add percentages of QATCH- generated data (10%, 20%, 30%, etc.) to the Spider dataset and retrain the models.
  3. Evaluate the performance of each of the fine tuned models on SpiderDev to assess the impact of QATCH data on the model.
- Batch size: 8 (recommended: 32; limited GPU memory; gradient accumulation is time-consuming)
- Learning rate: **5e-5**TAPAS/ **5e-3** TAPEX, reduced if validation loss plateaus
- Training time: 5 epochs per experiment
- Sample and split data in a stratified way based on SQL tag.
- At each run, provide more data in the category where performance is worse.

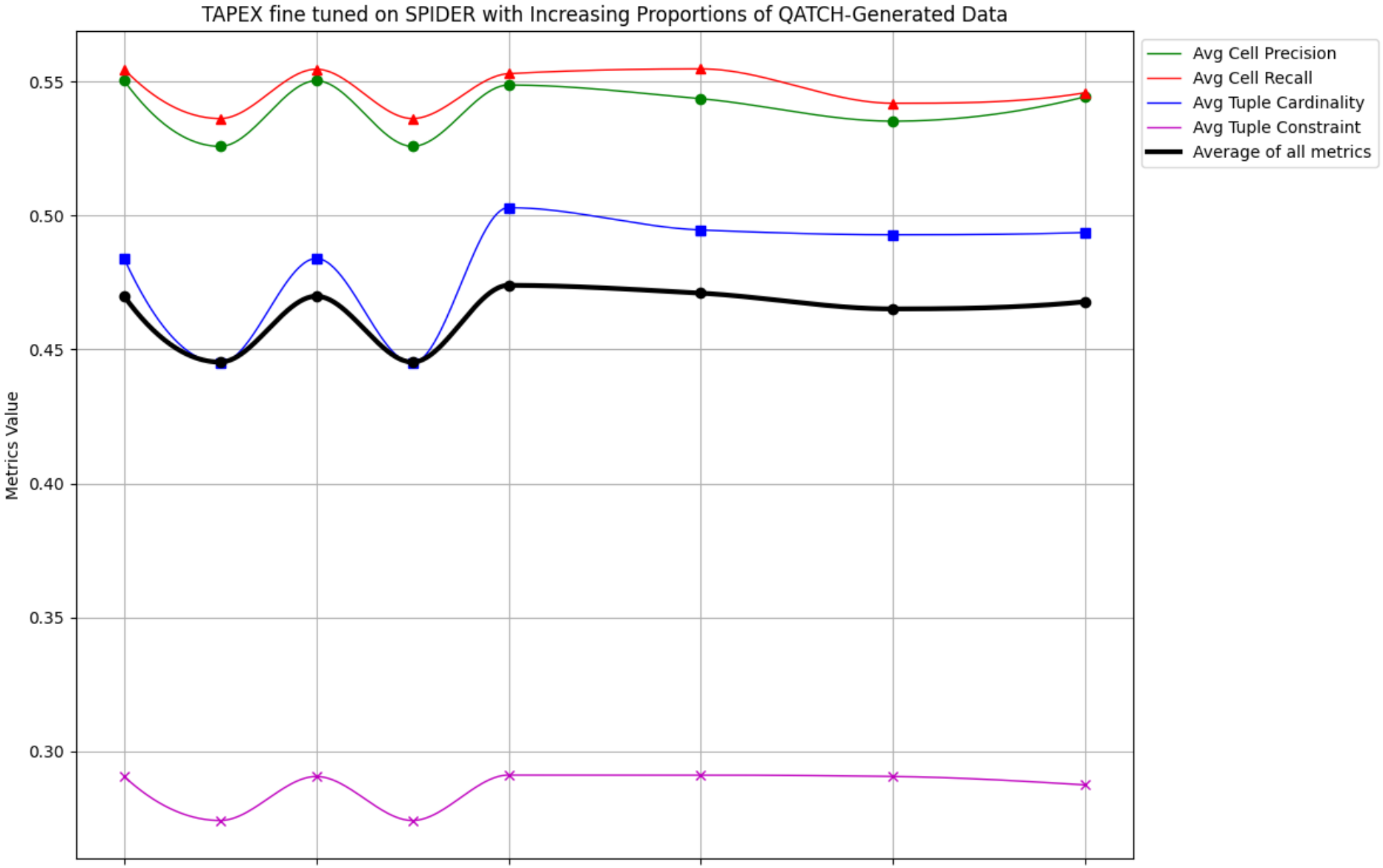
Results TAPAS: Spider + QATCH synthetic data (no LLM var)



Interpretation

- Tuple constraint and recall peak at 30%; 80% and 100% show no increase in these metrics but inflate tuple cardinality and precision.
- A small percentage (30%) enhances task understanding; more data increases precision but not task understanding.
- We also noticed a gap between validation and testing results. The model could be overfitting occurs due to simple, template-based QATCH NL questions.

Results TAPEX: Spider + QATCH synthetic data (no LLM var)



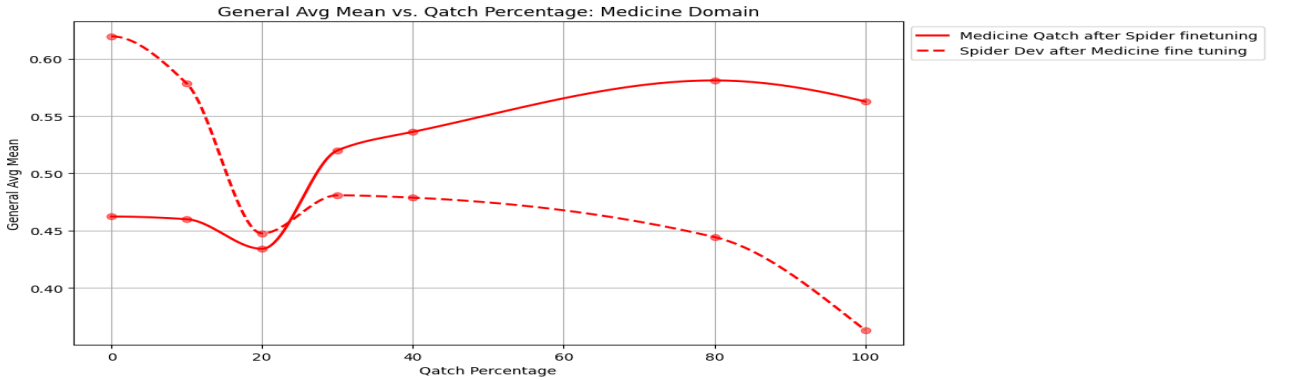
Interpretation

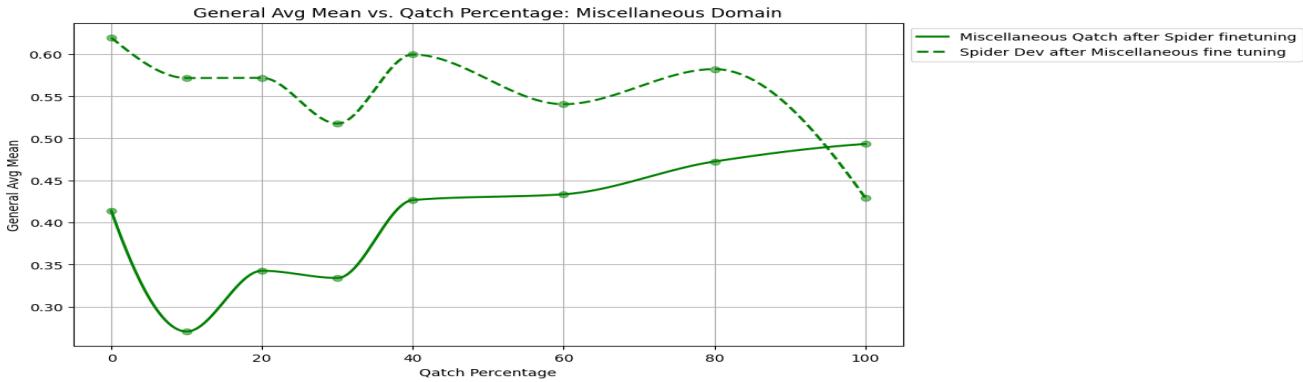
- initial performance drop, then stabilization, never exceeding base performance on Spider.
- TAPEX trained on a smaller corpus (0.5M vs. TAPAS 21M); QATCH examples may not align with high-quality training data.(eventhough the neural SQL engine was trained using high quality template queries as well)
- Experiments are time-consuming, each data point takes 30-40 minutes on a T4 GPU.

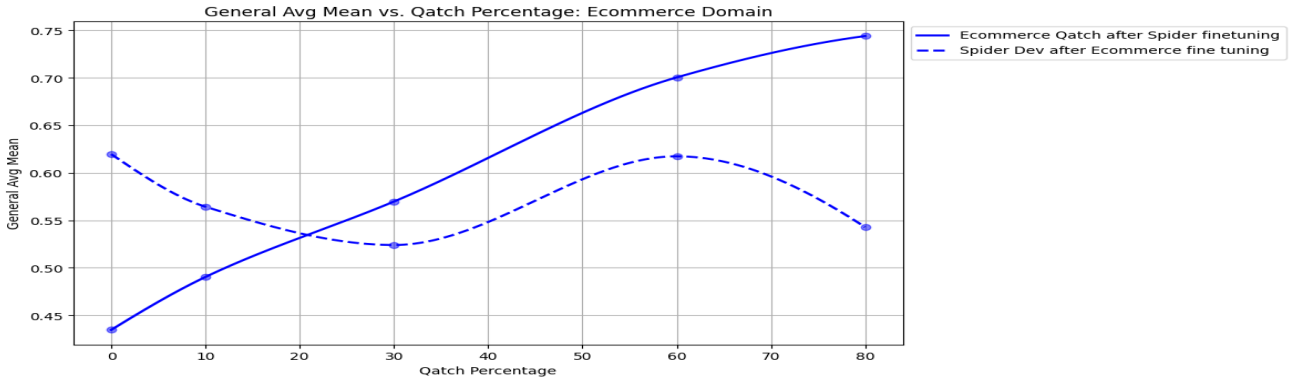
Experiments: Fine-tuning TAPAS on proprietary data(no LLM var)

- Determine if domain-specific fine-tuning affects the performance on the original generic multidomain datasets and identify any tradeoffs:
  1. Take TAPAS fine tuned on spider as a base Model (0% QATCH data).
  2. Fine tune the model at diffrent percentages of QATCH- generated data (10%, 20%, 30%, etc.) on domain specific data (Medicine, Finance, ...).
  3. Evaluate the performance of each of the fine tuned models on SpiderDev and on the retained domain specific QATCH data.
- Same hyperparameters as before.
- 400 examples approximately in each domain 400 rows, split into training, validation, and testing with 70/15/15.
- Take results with a grain of salt because of data leakage problem and a small testing sets.

Results


General Avg Mean vs. Qatch Percentage: Medicine Domain
General Avg Mean
Qatch Percentage
Medicine Qatch after Spider finetuning
Spider Dev after Medicine fine tuning


General Avg Mean vs. Qatch Percentage: Miscellaneous Domain
General Avg Mean
Qatch Percentage
Miscellaneous Qatch after Spider finetuning
Spider Dev after Miscellaneous fine tuning


General Avg Mean vs. Qatch Percentage: Ecommerce Domain
General Avg Mean
Qatch Percentage
Ecommerce Qatch after Spider finetuning
Spider Dev after Ecommerce fine tuning

Interpretation

- increased QATCH-domain specific data performance with more data (outliers/oscillations removed for clarity); significant improvement across domains noted.
- Performance on Spider data decreases with more QATCH data.(10% is capable of making the model forget)

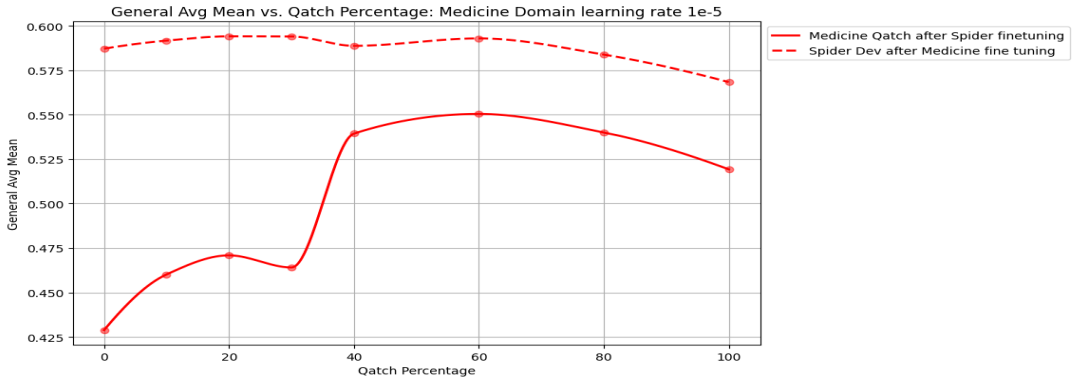
Control Experiment: Why does Performance on Spider drop?

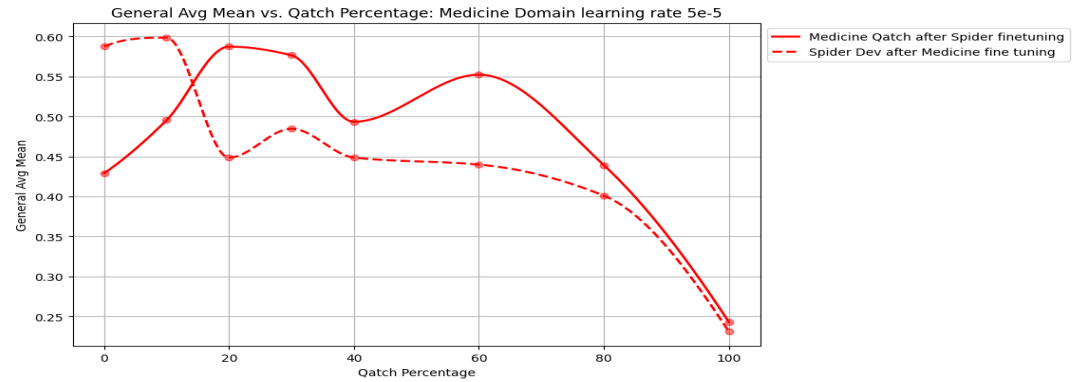
- We isolate three factors that could impact this issue:
  1. The style of the queries is simple and doesn't resemble the Spider style $\Longrightarrow$ Fine tune on spider style LLM vairations.
  2. The learning rate might be too high, so we train the model with a lower learning rate(divided by 5).
  3. We mix the QATCH data with some Spider data to remind the model of its previous training.(not seen in training e.g validation)
- Same hyperparameters as before.
- Draft/Trial and error experiment $\Longrightarrow$ Use results in the next experiments.
- Result: **The only factor that made a significant improvement was the learning rate.**

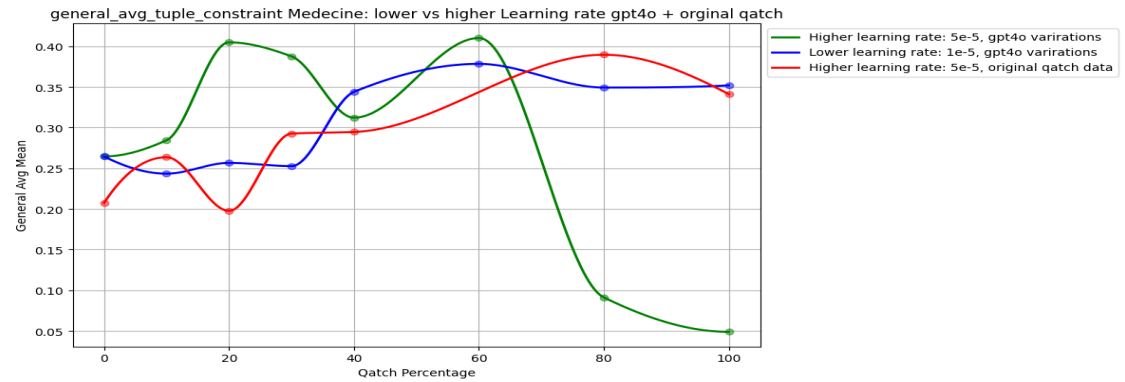
Fine-tuning TAPAS/TAPEX on LLM variations of proprietary data

- Training and testing models on simple/template-based QATCH data don't reflect true performance. How will models perform if trained on complex/realistic LLM variations of QATCH domain-specific questions?
  1. Take TAPAS/TAPEX fine tuned on spider as a base Model (0% QATCH data).
  2. Fine tune the model at diffrent percentages of GPT-4o variations of QATCH- generated data (10%, 20%, 30%, etc.) on domain specific data (only Medicine).
  3. Evaluate the performance of each of the fine tuned models on SpiderDev and on the original domain specific QATCH data(not llm var).
- divide learning rate by 5
- Same hyperparameters as before. 560 examples

Results


General Avg Mean vs. Qatch Percentage: Medicine Domain learning rate 1e-5
Medicine Qatch after Spider finetuning
Spider Dev after Medicine fine tuning
General Avg Mean
Qatch Percentage

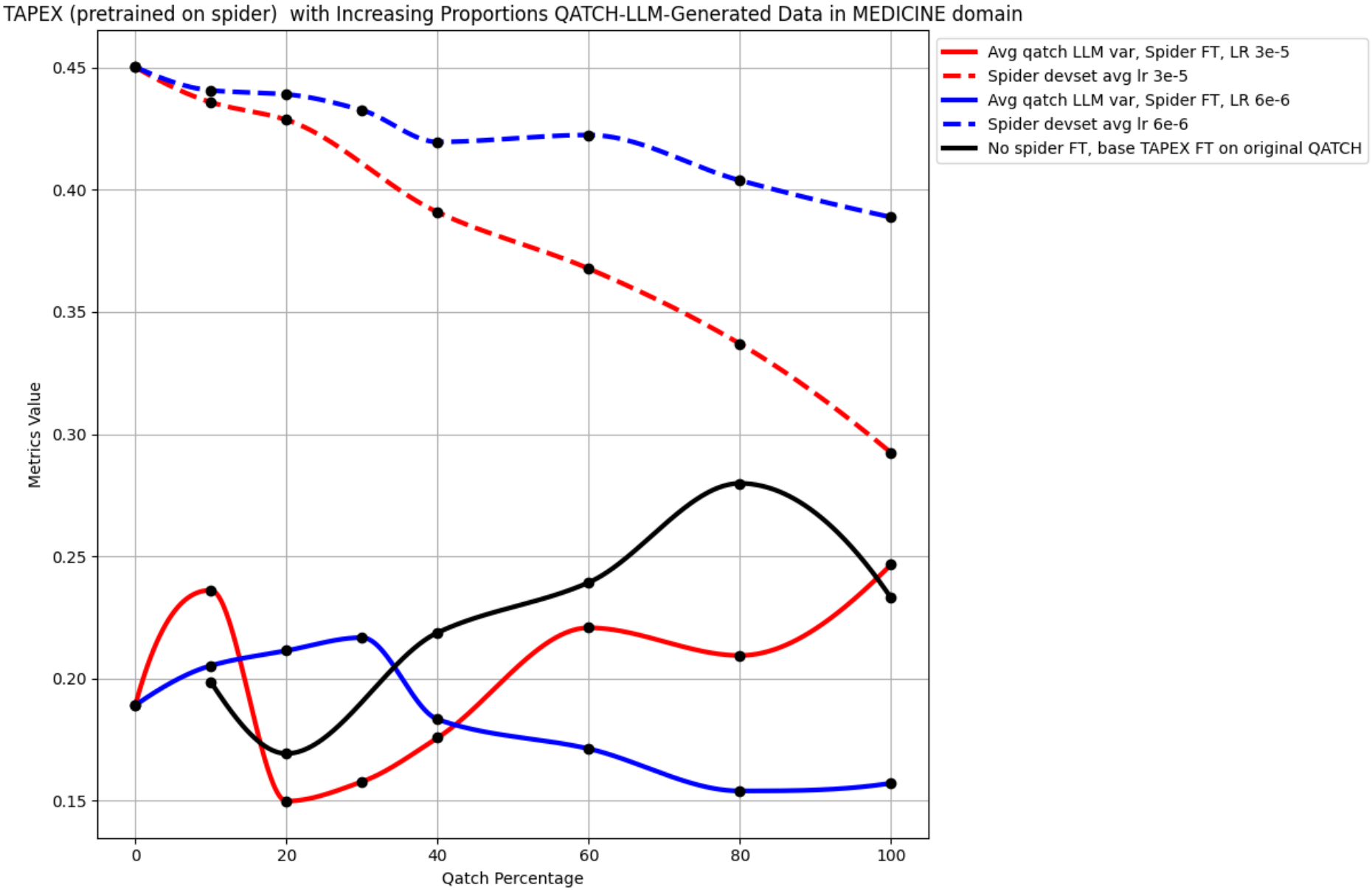

General Avg Mean vs. Qatch Percentage: Medicine Domain learning rate 5e-5
Medicine Qatch after Spider finetuning
Spider Dev after Medicine fine tuning
General Avg Mean
Qatch Percentage


general_avg_tuple_constraint Medecine: lower vs higher Learning rate gpt4o + orginal qatch
Higher learning rate: 5e-5, gpt4o varirations
Lower learning rate: 1e-5, gpt4o varirations
Higher learning rate: 5e-5, original qatch data
General Avg Mean
Qatch Percentage

Interpretation

- A learning rate of 1e-5 achieves good QATCH performance but drops with more LLM data due to saturation (five LLM variations per question).
- Normal learning(5e-5) rate achieves high initial QATCH performance, peaking at 20%, but Spider dev performance plummets quickly.
- Tuple constraint almost doubled!
- Keep in mind that we are working with the smaller versions of the models benchmarked in the QACTH paper.

TAPEX Results



Interpretation

- TAPEX responds well to these complex examples and doesn't get saturated when the learning rate is high.
- Even with a lower learning rate Spider performance isn't stable.
- Tuple contraint compared to the paper improved from **0.0** (TAPEX-LARGE-WTQ in the paper) to approximately **0.1**, and recall and precision improved from **0.04** and **0.37** to **0.36** and **0.33**, respectively. (our test set contains 63 examples).

Conclusion

- QATCH can boost performance of models on generic and proprietary datasets.
- Tradeoff: higher learning rate improves new workload performance but worsens performance on older ones.
- Model background is crucial to curate fitting data. Too many variations can saturate some models.

Future work

- Use more recent complex multidomain datasets such as Bird Li, J., et al. (2023). Can LLM already serve as a database interface? A big bench for large-scale database grounded text-to-SQLs.) A Big Bench for Large-Scale Database Grounded Text-to-SQLs
- Use more capable models which don't impose the limits of 512/1024 tokens.
- Use an ensemble model of SOTA TextToSQL systems to check the reverse mapping.
- Generate only 1 to 2 variations of the same question not to saturate the models.

Thank You For Your Attention!

Appendix: GTP prompt for variations

- "You are a helpful assistant that assists the user in generating variations of the question."
- " Given a SQL query, schema, table name, and a SQL Tag containing operation, and condition (if any),"
- " return at least 5 possible question variations in natural language of the sql query in JSON format."
- " Do not use the exact words in the query such as columns or other specific terms; separate them if they are stuck together. It has to be natural, informal language where we simulate that we don't really know the labels but have a vague idea, just like the way a human would formulate the questions without referencing the exact same labels but something close to them."
- " Each item in the JSON output should be a dictionary with the key as the question number and the value as a list containing two elements:"

Appendix: GTP prompt for variations

- ” the first element should be the natural language question, and the second element should be a minimalistic description.”
- ” The minimalistic description should follow strictly the following template: operations (there could be many), target columns (there could be many), target table.”
- ” The minimalistic description should only use words from the original question.”
- ” The minimalistic description shouldn’t contain filler words because it is used in embedding cosine similarity, so a single word can skew the similarity; be mindful.”
- ” The minimalistic description and the generated natural language questions should only use words from the original question except for alphanumeric and non-letter characters in words which should be transcribed to English.”