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

Moaad Maaroufi[1], Paolo Papotti[1], Luca Cagliero[2], Simone Papicchio[2]

[1]EURECOM Sophia Antipolis, France
[2]Politecnico di Torino Turin, Italy
{maaroufi,papotti}@eurecom.fr
{cagliero,papicchio}@polito.it

Abstract

"Today's language models cannot 'read tables' reliably" Li et al. (2023b). Most models are pre-trained on 1D natural language, whereas tables are 2D, which is probably the cause of their unreliability. The work on QATCH (Query-Aided TRL Checklist) (Papicchio et al., 2023) shows the necessity of training Table Representation Learning (TRL) systems on examples that include the target table. Nevertheless, performance degrades when the model is used on a new, unseen dataset, especially if it is from a very specific domain, which is rare in training data.

To overcome this problem, we need to produce a collection of training data in the form of pairs of NL question and table with answer, which is infeasible to do manually. This is where the need for automation comes into play.

We need to generate a set of high-quality training examples for a given user-proprietary dataset, unseen in the training data of the TRL model, i.e., diverse in terms of SQL and NL, and reflective of the user workload if available.

Previous work shows that synthetic training examples can be derived automatically from structured data to train NLP-based fact-checking systems (Veltri et al., 2023; Bussotti et al., 2024) and for benchmarking Text2SQL systems (Papicchio et al., 2023). In this work, we try to answer the following questions:

- Does fine-tuning with synthetic QATCH examples improve Table Question Answering (TAPAS, transformer architecture)?(TAPAS Herzig et al. (2020))?
- Can we get comparable results across different models (TAPEX Liu et al. (2022))?
- Does fine-tuning domain-specific data break the model's (TAPAS) performance on the original generic dataset (Spider Yu et al. (2019))?
- Training and testing models on template based simple QATCH data don't reflect true performance. How will models perform if trained on complex/realistic LLM variations of QATCH domain-specific questions?

1 Introduction

The scope of this study is to help assess TRL models on proprietary data in only one of two established SQL-centric tasks: 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 table schema), building on (Papicchio et al., 2023). The motivation for TRL, as explained in (Papicchio et al., 2023), stems from the fact that models trained using this framework are able to surpass the limitations of traditional declarative specifications based on first-order logic and SQL. We focus only on QA in this work for three reasons:

- We want to expand on the results of (Papicchio et al., 2023). Fine-tuning SP models benchmarked in the paper is beyond the compute power allocated for this project, and using a quantized version wouldn't be true to the paper. Some QA models are much more accessible for single GPU training. One could think about gradient accumulation, but it would take too much time, and consequently, we cannot perform as many experiments as we do in our case.
- In (Li et al., 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, Column Type Annotation, Schema Matching, Entity Matching, Missing Value Identification, Column Finding, Table Question), and 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.

For the above reasons, we use TAPAS (Herzig et al., 2020) and TAPEX (Liu et al., 2022) throughout our experiments, with a slight inclination towards TAPAS.

Before tackling the main problem, we start with some warm-up experiments to establish a baseline in order to prove that models can benefit from QATCH data to boost performance on a generic dataset(Spider (Yu et al., 2019)). It consists of fine-tuning TAPAS and TAPEX on SPIDER plus increasing amounts of synthetic data generated by QATCH from SPIDER tables.

The next experiment determine if domain specific fine tuning affects the performance on the original generic multidomain datasets and identify any tradeoffs. We do this by using QATCH to generate synthetic data on the domain-specific proprietary Kaggle tables used in (Papicchio et al., 2023) and fine-tune the previously trained model(TAPAS only on Spider) with increasing amounts of synthetic data, just like before.

The QATCH queries are quite simple, and may not reflect true performance. We use some LLM like GPT-4o to reformulate the questions and introduce some realism and complexity(the user doesn't know the exact schema). The models (TAPAS, TAPEX) are trained exclusively on LLM variations and tested on the original template based data. We discuss the conservation reverse mapping NL $\Longrightarrow$ SQL as well given an LLM variation of a SQL query. We explain the findings throughout these experiments.

2 Datasets

Throughout our experiments, the SPIDER dataset (named so because it is cross-domain and complex) is present in one way or another. It consists of **10,181** questions and **5,693** unique complex SQL queries on **200** databases with multiple tables covering **138** different domains (Yu et al., 2019).

SPIDER queries touch on multiple tables and use operations like EXCEPT and JOIN, which neither QATCH nor TAPAS/TAPEX support. Thus, we need to do a lot of cleaning and filtering. While QATCH source code already performs some of this filtering, it does not take into account specific requirements of TAPAS and TAPEX. For example, in aggregation, the query should perform only aggregation and not selection. For instance, the query:

```
SELECT C1, MIN(C1)
FROM table
GROUP BY C1
```

is not supported by TAPAS; it should instead be

```
SELECT MIN(C1)
FROM table
GROUP BY C1
```

Therefore, the cleaning process is developed from scratch to better suit the needs of these two specific models and to have more control over the pipeline. This is detailed in a future section.

We also use some proprietary data that the models haven't seen before, introduced in (Yu et al., 2019) as Kaggle datasets. These include 8 tables, 2 for each domain: Medicine, E-commerce, Finance, and Miscellaneous. They also go through some preprocessing supported by QATCH. We detail this preprocessing in the following section as well.

2.1 Preprocessing

2.1.1 datasets: Spider

It took some time to build a reliable data preprocessing pipeline for TAPAS and TAPEX.

First, we download the Spider dataset objects from Hugging Face Hub, which provides the training set and the dev set with natural language questions and their SQL equivalents, along with the tokenized queries, NL questions, and the database IDs. The actual database tables are not provided in the dataset and should be downloaded from the official SPIDER webpage.

Next, we need to remove queries that touch on more than one table, as the models we use don't support this. We do this by taking the database ID that comes with each query/question, accessing the database to get the table names present, and matching them to the SQL query using regex. If we find that more than one table is used, we remove that query. There is still a problem: some queries perform table self-joins, which we also don't support. We will address this issue as well.

Queries that use tables containing more than 15 rows should also be filtered. This is because TAPAS and TAPEX(Weak supervision) require us to provide the actual table along with the NL question and the answers. All of this should fit within 512/1024 tokens. The choice of 15 rows was determined through experimental trial and error, and we tuned for this value. One could verify the token constraints after gathering all the components for the input, but it would be a waste of resources as most tables with more than 15 rows tend to surpass the 512/1024-token mark. Therefore, it's more efficient and uniform to remove them from the beginning.

At this point, we filter out the queries to remove the ones using these operations: 'join', 'union', 'intersect', 'except', 'inner', 'outer', 'left', 'right', 'full', 'limit', 'like'. Additionally, we check for nested queries by counting the occurrences of 'select' and remove them. We also ensure there is at most one aggregation keyword: 'count', 'sum', 'avg', 'min', 'max'.

TAPAS takes pandas dataframes as inputs instead of SQLite tables. We first get the tables and put them in a dataframe. Next, we fetch the answers to the queries by executing each query on its corresponding database. If we are giving answers we have to TAPAS, the HuggingFace training pipeline forces us to give coordinates of these answers as well as lables for loss calculation. By coordinates, we mean that the top-leftmost cell represents (0,0). For this purpose, we need to establish some kind of order which was absent in the SQLite databases. This is a very tricky process because, even if we have the answers, mapping them back to the table is pretty much impossible.

This is because sometimes the mapping can be ambiguous when we have duplicate answers, or when we have aggregation, it can be challenging to determine which cells are needed to perform the aggregation.

For this purpose, we will use a simplified approach. We modified a script provided by the Tapas repository to locate the answers and report the coordinates of the first occurrence each time. For aggregation, a single value is returned sometimes and multiple values at other times. The script can detect single aggregation values, but it fails to identify and handle multiple aggregation values. For this reason, the latter are discarded.

TAPEX requires answers, natural language questions, and an ID for each row. These should be passed in JSON format. It's relatively easier than preprocessing for TAPAS, which requires answer coordinates as well.

Once all the training components are ready, we split the dataset into 80% training and 20% validation for the Spider dataset. We use the dev set as the test set. Note that the number of examples drops drastically after all the cleaning, from **7000** to **1400** in the training set and from **3000** to **200** in the dev set.

To train the models, we need to tokenize the data using the appropriate tokenizers for each model. A significant constraint observed in (Papicchio et al., 2023) is that TAPAS imposes a 512 token limit, while TAPEX allows 1024 tokens on table plus question as input (no issues with TAPEX during training). The tokenizer will handle this for us; it will trigger an error whenever we encounter such a problem, which we will catch and then discard the problematic example.

2.1.2 datasets: QATCH generated Data

QATCH takes in a SQLite database or a pandas dataframe as input (Spider/Kaggle proprietary data) and generates natural language questions and correct SQL queries based on a template for different categories. However, we cannot control the number of examples generated for each category or in total. For Spider, we only provide QATCH with the databases used by the training examples, not the dev set examples. This results in 21,000 questions and SQL queries, but we will not use all of them. Similarly, we apply the same process to the Kaggle proprietary tables after truncating them to 15 rows. The queries output by QATCH undergo the same cleaning process as before to ensure compatibility with the models. We get 400 examples for domain approximately.

2.1.3 Datasets: QATCH NL variations using GPT4o

The QATCH natural language questions are built on very simple templates, which increases the risk of models overfitting to them (we will discuss this further in the experiments section). To mitigate this, we can use a large language model (LLM) with few-shot learning to create different variations of the same question, using synonyms and different wording. We initially experimented with GPT-3.5-turbo, and once we were satisfied with the results, we finalized with GPT-4o.

There are a lot of problematic questions that need to be answered here. First, how "complex" should the questions be? Should the questions assume that the user knows the exact schema, as the Spider and QATCH-generated questions do? How much should we assume that the user knows? Furthermore, should we consider the temperature as a hyperparameter, or is prompt engineering enough? What should we provide to the LLM: the QATCH natural language questions or the SQL queries directly?

The last and most important question that should be addressed is how we can ensure that the mapping between the LLM-generated natural language (NL) question and the original SQL query is maintained.

We address all of these questions one by one, presenting the adopted solutions and their compromises, starting in this section and continuing in the experiments section. The choices are tightly correlated with the experiments and the feedback we received.

We start with the mapping question: to guarantee the mapping, we need to map the natural language question back to SQL. This is especially important when we don't use the exact names in the schema. One way to do this is by using an ensemble of state-of-the-art models to verify whether they agree on the equivalence with the original query. However, this approach came up very late in the project's duration, making it infeasible. Additionally, it is very compute-intensive to perform this verification for every single variation, given that we need to generate multiple variations for each question, even though it is the most trustworthy approach.

The second option is to use OpenAI embeddings on the original QATCH NL question and the LLM-generated ones, setting a threshold for cosine similarity. This approach is straightforward, but even with the most capable OpenAI text-embedding-3-large model (ranked 19th on the Massive Text Embedding Benchmark (MTEB) Leaderboard (Muennighoff et al., 2023)), the results are overoptimistic and not very reliable.

We noticed that when the two questions have 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.

To address this, we could ask the LLM to provide a minimalistic description of the generated question following a template: operations, target columns, and target table, using only words from the generated questions. We could then average the embeddings of both the generated question and the minimalistic description. However, the minimalistic description is flawed because it loses the order and important context; the goal was to remove filler words to achieve an unbiased score.

After inspecting some of the generated questions, we found that they are of very good quality. Therefore, we decide to adopt the following approach: We do not check the mapping at this stage; instead, we leave it to the testing phase to decide i.e if the performance after fine tuning drops then the mapping is bad, otherwise if the model generalize to the simple QATCH templates when exclusively trained on the LLM variations we can infer that the mapping is conserved for most examples. We train the model on the LLM-generated questions only. Then, we test it on the original QATCH questions, which is simple and known to be correct. If the model performs well, we can infer that it can generalize and that the mapping is correct most of the time. The questions that don't conserve the mapping will act as regularizers.

Examples:

- 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?

We do not relinquish all control to the LLM in generating the varied queries. Instead, we impose some constraints by providing the LLM with the schema and the SQL query (only the SQL query and not the QATCH NL question). This is because 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.

Additionally, we provide the schema and the SQL tag of the query generated by QATCH. Furthermore, we will ask the LLM to still provide the minimalistic description, but not for embedding purposes. Instead, the description will serve to make the LLM aware of the importance of the embedding. By doing this, the LLM will try to maintain the necessary keywords in the generated question, knowing that it will have to provide them in the minimalistic description.

For the temperature, we leave it at the default (0) to make the LLM follow the instructions closely. We still get very diverse queries thanks to prompt engineering, and adding another hyperparameter would complicate understanding the models' performance.

Regarding the "complexity" of the questions, we experiment with two approaches. The first approach generates queries in the same style as Spider, using the exact schema and a wording style similar to Spider by providing the LLM with some examples of Spider NL questions and SQL queries. This experiment is a brief control experiment. The prompt instructs GPT to include the exact schema and to generate questions in the same style as spider by providing it with 15 NL question and their SQL equivalents.

We are more interested in nuanced questions with synonyms, assuming that the user doesn't know the exact names of the schema. This approach aligns closely with SOTA datasets like (Li et al., 2023a). This would enable us to see if the models can generalize to simple QATCH questions after being trained exclusively on these "complex" questions. The prompt here on the other hand instructs the model to suppose that the user can only describe the schema in a human way and doesn't know the exact names and provide it with similar examples of the desired output.

3 Experiments

Spider Experiment The initial experiment establishes a baseline TAPAS performance on the Spider dataset as has been explained before. We then aim to improve this performance using QATCH-generated data derived from Spider tables.

Our approach involves the following steps:

- Train the model on the Spider dataset (0% QATCH data).
- Incrementally add percentages of QATCH-generated data on spider tables (10%, 20%, 30%, etc.) to the Spider dataset.
- For each percentage, retrain the TAPAS model from scratch using the combined Spider and QATCH data.
- Evaluate the performance of each retrained model to assess the impact of QATCH data on the model.

Only in this experiment, we train from scratch. For the experiments in the next sections, we take the fine-tuned model on Spider and use it as the base model. This was a misunderstanding, and it was too late down the road of the project to rerun the experiments due to time and cost constraints. As you may notice, we can train the model only once on Spider instead of doing it multiple times. However, this does not take any credibility from the results, because we will see if adding more batch data increases performance or not.

We should note that we are using a batch size of 8(recommended batch size is 32, but not enough GPU memory, accumulating gradient is time-consuming), we use a learning rate of **5e-5**, which is reduced on plateau of the validation loss(the Cell Selection Loss and Scalar Answer Loss in case of aggregation) for 2 epochs by a factor of 0.1. For each of these experiments the model was trained for 5 epochs.

After each experiment, we assess performance using QATCH metrics on our development dataset. We try to improve performance in the weakest categories by adding relevant examples. However, QATCH's tendency to generate questions biased towards "HAVING" clauses often results in a shortage of examples in other categories, limiting our ability to fully address all performance gaps. When this happens, we are forced to complement with additional examples from the overrepresented categories, even if they are not directly relevant to the weaker areas, simply to reach the desired percentage.

TAPAS Figure 1 shows the results of these experiments with TAPAS. The performance across all metrics is better than the starting point. Focusing on the tuple constraint (arguably the most important metric because it shows if the model respects the format of the true answer, i.e., understanding of the task) and recall, we see that it peaks at 30%.

The 80% and 100% (unrealistic and dilutes Spider data, which we don't want in real life) show no increase in tuple constraint or recall but do show a significant increase in tuple cardinality and precision (these two metrics inflate the overall average score).

This indicates that a very small percentage, 30% in this case, is capable of increasing the model's understanding of the task (tuple constraint). More data increases precision but doesn't enhance the understanding of the task. We can explain this by the fact that the model is overfitting, as there is a considerable difference between the validation and testing results as we increase the QATCH percentage. This overfitting occurs because QATCH NL questions are simple and template-based. This issue will be addressed in the following sections.

TAPEX We performed the exact same experiment with TAPEX, using similar hyperparameters (batch size 8, learning rate 3e-5, the default one). Figure 2 summarizes the results.

Initially, there seems to be a drop in performance, but then it stabilizes, never exceeding the base performance on Spider across all metrics. This could be explained by the fact that TAPEX was trained on a much smaller corpus compared to TAPAS, which used very high-quality examples (TAPAS 21M vs. TAPEX 0.5M, based on the TAPEX.md Hugging Face GitHub). The QATCH examples are very simplistic and may not align with this high-quality training data.

Running these experiments is time-consuming, with each data point on the graph representing an experiment of 30 to 40 minutes on a T4 GPU.

Fine-tuning TAPAS on proprietary data We aim to determine if domain-specific fine-tuning affects the performance on the original generic multidomain datasets and identify any tradeoffs.

We start by taking the previously fine-tuned TAPAS model on Spider(0% QATCH) and train it incrementally with domain-specific data generated by QATCH, just like before. We train and test on the Qatch data, which could be considered data leakage since the Qatch data uses the same template, so the performance should be taken with a grain of salt. We use the same hyperparameters as before. The generated Qatch data for each domain (2 tables of 15 rows) is approximately 400 rows, which is split into training, validation, and testing with 70/15/15 percentages, respectively. We report the performance on the QATCH test set and on the Spider dev set to see if the model forgets its prior training. Please note that the learning rate is still 5e-5 for TAPAS. We will only conduct this experiment for TAPAS and not TAPEX. The figure 3 shows the results.

We observe that performance on the QATCH data increases as more data is provided (some outlier points/oscillations have been removed for better visibility of trends). Compared to (Papicchio et al., 2023), the performance jump across the different domains is quite significant. However, this should be taken with a grain of salt due to the data leakage problem.

On the other hand, it is important to note that the performance on the Spider data decreases as we provide more QATCH data.

Fine-tuning TAPAS on LLM variations of proprietary data Before conducting the next experiment, we investigated the problem of performance degradation on the Spider dev set. We isolated three factors that could impact this issue:

- The style of the queries is simple and doesn't resemble the Spider style. To address this, we ask the LLM to generate questions in the same style as Spider by providing some examples and then trained the model this way.
- The learning rate might be too high, so we train the model with a lower learning rate(divided by 5).
- We mix the QATCH data with some Spider data to remind the model of its previous training.

These were trial-and-error draft/control experiments, where we added and removed different elements. We found that the only factor that made a significant improvement was the learning rate. If it is too high, the model starts to forget. If it is low enough, we can achieve good performance on both the Spider dev and QATCH data. However, there is a tradeoff to be made, as we will see in the next experiment.

In the main experiment we use the LLM data on a specific domain (medicine), the LLM was given 112 qatch SQL queries and asked to generate 5 question variations for each SQL query which amounts to 560 questions in total. We contained the LLM generation process by guiding the LLM on how to generate the query to maintain the mapping

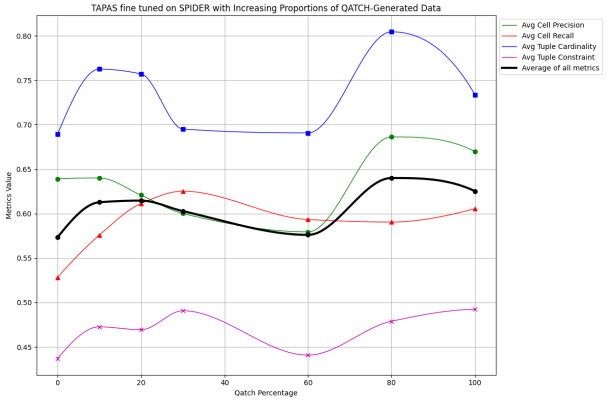

Figure 1: QATCH Metrics vs QATCH percentage: TAPAS fine tuned on SPIDER with Increasing Proportions of QATCH-Generated Data

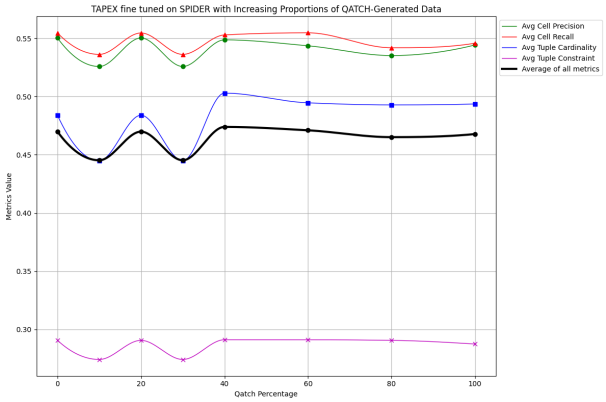

Figure 2: QATCH Metrics vs QATCH percentage: TAPEX fine tuned on SPIDER with Increasing Proportions of QATCH-Generated Data

as much as possible, and use temperature zero so that the LLM follows the instructions closely.

The results are encompassed in the set of graphs in Figure 4, which demonstrate how changing the learning rate plays a crucial role for TAPAS.

The first plot shows the mean of all metrics on the test set (composed of the original simple Qatch questions) and on the Spider dev set (the model was trained exclusively on LLM data after Spider FT). It demonstrates that with a learning rate five times smaller (1e-5) than the previous learning rate, we achieve good performance on the Qatch data. However, performance drops as we increase the percentage of the LLM data. This could be attributed to the model being saturated by two factors: generating five LLM variations for each question and Qatch itself generating multiple variations of a single question.

The second plot shows that with the normal learning rate used in the previous experiments, we achieve much higher performance on the Qatch data initially, but it starts to drop quickly. Interestingly, we get peak performance at only 20% of 560. The catch here is that the Spider dev performance also plummets quickly.

The third graph serves as a reminder of the performance of TAPAS on Qatch data when it was trained exclusively on Qatch data with no LLM variations.

The fourth figure concatenates all the plots of the performance on Qatch data into one graph for comparison.

The last figure shows the tuple cardinality across all experiments for comparison and demonstrates that we can actually double it using high-quality LLM data. This is a very good metric to check the models' understanding.

Fine-tuning TAPEX on LLM variations of proprietary data We perform the same experiments with TAPEX in this section.

We also use a batch size of 8 and a learning rate of 3e-5, which is divided by 5 to get 6e-6, just like with TAPAS.We take the previously fine-

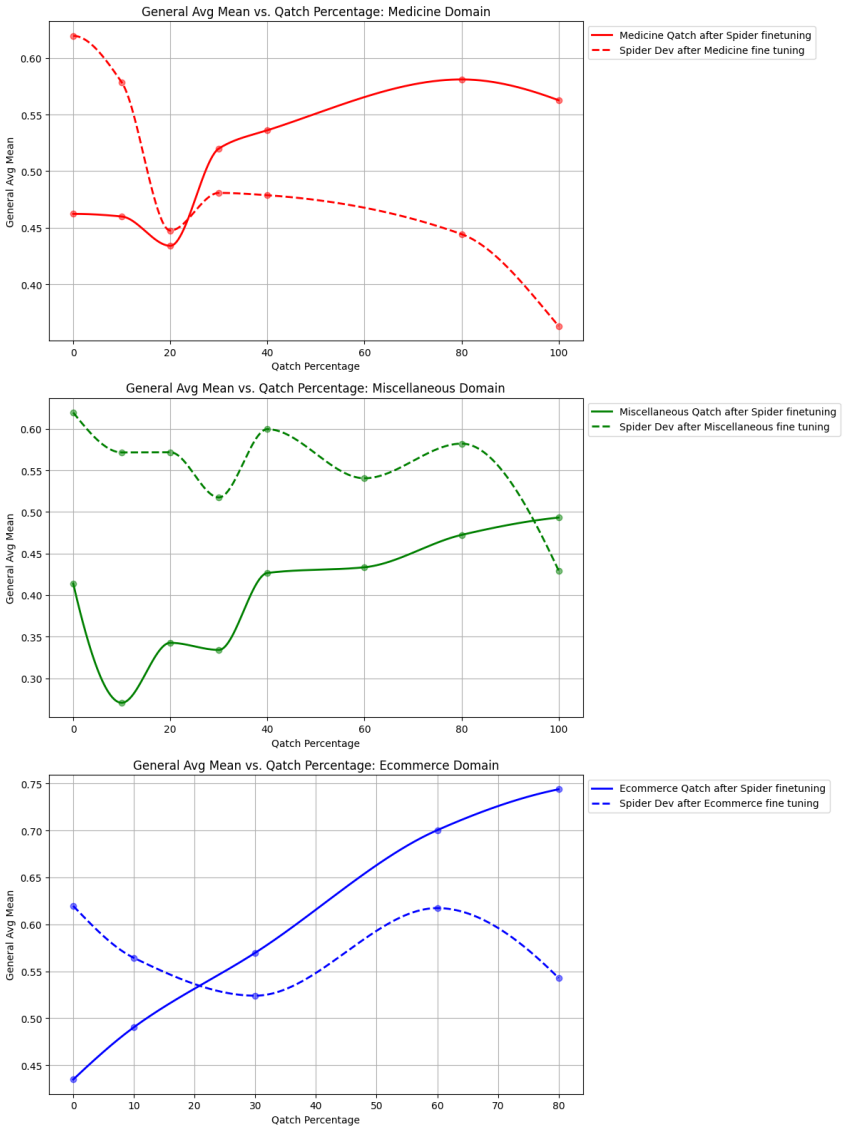

Figure 3: QATCH Metrics vs QATCH percentage: TAPAS fine-tuned on Increasing Proportions of domain specific QATCH-Generated Data after FT on Spider

tuned TAPEX on Spider and then fine-tuned it on different percentages of the variated LLM data. It is trained for 5 epochs on each percentage.

The results showcased in Figure 5 are quite similar to TAPAS with some minor differences. The goal of this study is not to compare the performance of the models, but to observe the impact of QATCH and see if the trends in both models are the same.

The red and blue curves correspond to TAPEX fine-tuned on Spider and then on LLM-QATCH-varied queries but with different learning rates. The red curve represents a high learning rate of 3e-5, while the blue curve represents a lower learning rate of 6e-6 (divided by 5).

The continuous curves report the performance of TAPEX on the original QATCH data (the exact same questions that TAPAS was benchmarked on), which it had never seen as it was exclusively fine-tuned on varied LLM questions.

The dashed red and blue curves correspond to the performance on the Spider set as we gradually fine-tune on the QATCH LLM data.

The continuous black curve represents a control experiment performed with TAPEX base (not fine-tuned on Spider), trained on original QATCH queries. This is done to see if having seen the QATCH queries during training results in better performance compared to TAPEX, which had never seen these queries during training but had been fine-tuned on Spider and some varied queries. The results of this control experiment should also be taken with a grain of salt because, as explained

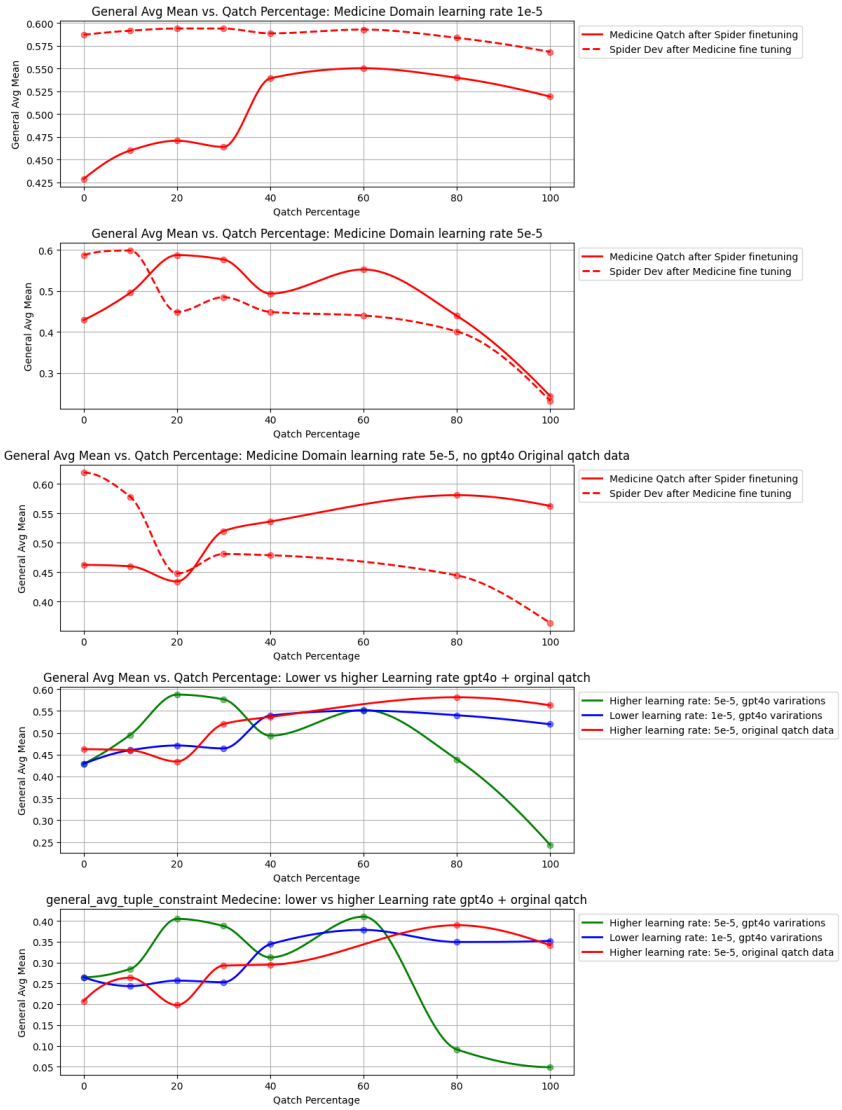

Figure 4: QATCH Metrics vs QATCH percentage: TAPAS fine-tuned on Increasing Proportions of domain specific LLM variations of QATCH Data after FT on Spider

earlier, there is a data leakage problem.

We observe the same trend where, with a lower learning rate, the performance on Spider decreases less aggressively. We also achieve better performance on QATCH when the learning rate is high. The difference here is that more data doesn't saturate the model when the learning rate is high. However, when the learning rate is low, more data can saturate the model. With a high learning rate, we achieve optimal performance with 100% of the data. Nevertheless, we manage to reach high performance even with just 10% of the data.

The metrics for TAPEX are generally much lower than TAPAS, as also noted in the paper (Papicchio et al., 2023) without any fine-tuning. Nonetheless, we notice that with the fine-tuning we performed, there is a performance bump across all metrics. Specifically, the tuple constraint 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. Here, we report only the mean of all metrics in each graph for conciseness, but the other graphs for each metric can be found on Colab.

4 Conclusion

In this study, we investigated if QATCH synthetic data is capable of improving the performance of some question-answering models. The answer is affirmative for TAPAS, but for TAPEX, the simple templates didn't suffice. This highlights the importance of understanding the background and pretraining of the model. TAPAS was trained on a

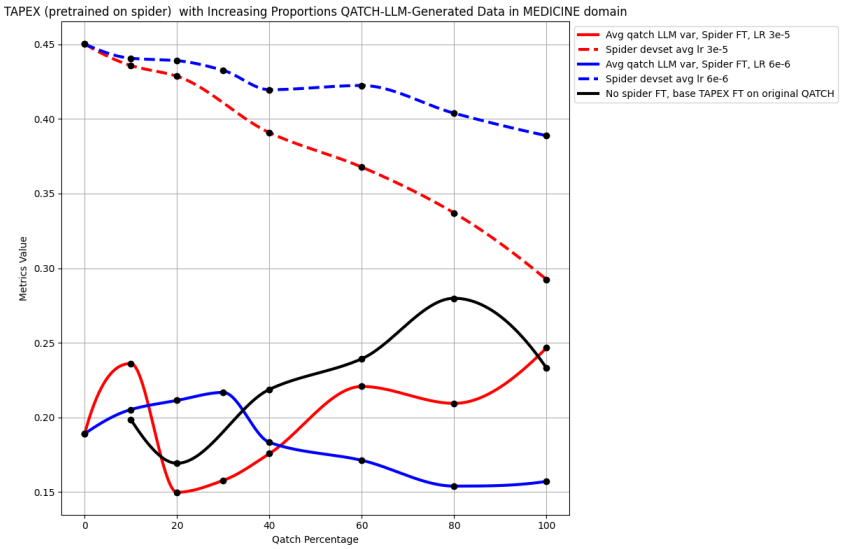

Figure 5: QATCH Metrics vs QATCH percentage: TAPEX fine-tuned on Increasing Proportions of domain specific LLM variations of QATCH Data after FT on Spider

noisy, large corpus with millions of tokens, while TAPEX was trained on only 0.5 million tokens, which are curated and high quality.

We observed that on proprietary data, higher learning rates tend to degrade the performance of models previously fine-tuned on generic datasets. We also noted that using QATCH LLM-varied queries, even without checking the reverse mapping from NL to SQL, managed to boost performance. We also noticed that too many variations can saturate some models, like TAPAS, but not TAPEX.

5 Future work

In future work, we could curate the data further by generating a maximum of one or two varied questions per QATCH query instead of five to avoid saturating the models and create more diversity. We should also check the mapping by using an ensemble of semantic parsing models.

Furthermore, TAPAS and TAPEX pose restrictions on the type of queries we can give them. Maybe we can work with Llama3 7b and use a modern dataset like BIRD Li et al. (2023a) instead of Spider to see if the model forgets its training, as BIRD has more complex and realistic examples than Spider.

QATCH can also benefit from adding fine-grained control over how many queries we want to generate per category using unique queries only. This is because providing multiple reformulations of a single query can eventually saturate the models, as we observed.

References

Jean-Flavien Bussotti, Enzo Veltri, Donatello Santoro, and Paolo Papotti. 2024. Generation of training examples for tabular natural language inference. In *SIGMOD/PODS 2024, ACM International Conference on Management of Data, 9-14 June 2024,¡br /¿ Santiago, Chile*, Santiago.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. Tapas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023a. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls.

Peng Li, Yeye He, Dror Yashar, Weiwei Cui, Song Ge, Haidong Zhang, Danielle Rifinski Fainman, Dongmei Zhang, and Surajit Chaudhuri. 2023b. Table-gpt: Table-tuned gpt for diverse table tasks.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. 2022. Tapex: Table pre-training via learning a neural sql executor.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. Mteb: Massive text embedding benchmark.

Simone Papicchio, Paolo Papotti, and Luca Cagliero. 2023. Qatch: Benchmarking sql-centric tasks with table representation learning models on your data. In *Advances in Neural Information Processing Systems*, volume 36, pages 30898–30917. Curran Associates, Inc.

Enzo Veltri, Gilbert Badaro, Mohammed Saeed, and Paolo Papotti. 2023. Data ambiguity profiling for the generation of training examples. In *ICDE 2023, 39th IEEE International Conference on Data Engineering, 3-7 April 2023, Anaheim, California, USA*, Anaheim.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2019. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task.