ASQP-RL Demo: Learning Approximation Sets for Exploratory Queries

Susan B. Davidson
University of Pennsylvania
USA
susan@cis.upenn.edu

Tova Milo
Tel Aviv University
Israel
milo@post.tau.ac.il

Kathy Razmadze
Tel Aviv University
Israel
kathyr@mail.tau.ac.il

Gal Zeevi
Tel Aviv University
Israel
galzeevi@mail.tau.ac.il

ABSTRACT

We demonstrate the Approximate Selection Query Processing (ASQP-RL) system, which uses Reinforcement Learning to select a subset of a large external dataset to process locally in a notebook during data exploration. Given a query workload over an external database and notebook memory size, the system translates the workload to select-project-join (non-aggregate) queries and finds a subset of each relation such that the data subset – called the approximation set – fits into the notebook memory and maximizes query result quality. The data subset can then be loaded into the notebook, and rapidly queried by the analyst. Our demonstration shows how ASQP-RL can be used during data exploration and achieve comparable results to external queries over the large dataset at significantly reduced query times. It also shows how ASQP-RL can be used for aggregation queries, achieving surprisingly good results compared to state-of-the-art techniques.

CCS CONCEPTS

• **Information systems → Data management systems**.

KEYWORDS

Queries approximation, Exploratory Data Analysis, Reinforcement Learning

ACM Reference Format:
Susan B. Davidson, Tova Milo, Kathy Razmadze, and Gal Zeevi. 2024. ASQP-RL Demo: Learning Approximation Sets for Exploratory Queries. In *Companion of the 2024 International Conference on Management of Data (SIGMOD-Companion '24), June 9–15, 2024, Santiago, AA, Chile.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3626246.3654741

1 INTRODUCTION

In the data exploration phase of data science, users execute a variety of queries, including both aggregate and complex non-aggregate select-project-join (SPJ) queries, often in an iterative manner [6]. Dealing with large databases and complex exploratory queries can be time-consuming, prompting users to prefer faster but approximated query results. Approximate Query Processing (AQP) [7] has been proposed to accelerate the processing of aggregate queries, but does not address complex non-aggregate SQL queries which are common in the data exploration process.

For example, a recent study of the IMDB Database [6] showed that in the query workload of data exploration sessions, only about 50% of the queries were aggregate queries. The remaining half were complex select-project-join queries. These complex queries had an average processing time of 25 minutes, and 30% of them exceeded 1 hour, even with result limits being imposed, making the wait time impractical. This demo extends the scope of approximate query processing to non-aggregate queries, presenting a solution that accelerates the execution of such queries with high accuracy, facilitating rapid insights in data exploration.

***Approximation of complex non-aggregates queries*.** Given a database and expected query workload which includes complex non-aggregate queries, our work focuses on (1) defining what a reasonable approximation is and (2) providing an efficiently computed approximation for queries. While approximate answers are well-established for aggregate queries, extending this idea to non-aggregate SPJ queries requires careful consideration. In Section 2, we discuss what an approximation for non-aggregate queries means, i.e. including some, but not necessarily all, of the tuples in the query result. The acceptability of this approximation is rooted in practical considerations: If the size of the query output exceeds what is humanly analyzable, users may not be able to discern that the result is approximated. Unlike AQP, where the quality metric for approximation is consistent across queries, the critical factor in non-aggregate queries is the size of the result that a user can visually analyze. In cases where the result is small enough to be analyzed by the user, the approximation must be highly accurate, encompassing the entire result set. As with aggregate queries, executing complex non-aggregate queries can be very time consuming. Drawing parallels to users' willingness to accept extended wait times for offline pre-processing in AQP algorithms to obtain approximate, but fast, query results, we posit that a similar scenario holds for algorithms capable of producing good approximate answers for non-aggregate queries. Before discussing our approach to handling approximate non-aggregate query processing, we outline the challenges and explain why existing work is not suitable.

This work is licensed under a Creative Commons Attribution International 4.0 License.

SIGMOD-Companion '24, June 9–15, 2024, Santiago, AA, Chile
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0422-2/24/06.
https://doi.org/10.1145/3626246.3654741

Challenges. Selecting an approximation set for subsequent querying presents various challenges: (C1) The vast combinatorial space, created by the number of combinations of tuples from different tables. (C2) Understanding the search space, due to the cost of executing the complete query workload. (C3) The varying size of query results, which introduces different importance levels for tuples, e.g. tuples from queries with smaller result sizes are more significant than ones from larger result sizes. (C4) Generalizing for future, potentially dissimilar queries, since the query workload only captures what is already known. (C5) Detecting a drift in users' interest.

Inadequacy of Existing Approaches. Although it might seem that existing solutions could address these challenges, they have several shortcomings. Online Analytical Processing (OLAP) requires significant storage resources for data cubes, in particular a nearly tenfold blowup of the original dataset size. OLAP also relies on proficient Database Administrators (DBAs) familiar with its unique syntax, limiting its use by general users. While AQP solutions seem promising, recent approaches using generative models (e.g., [4]) often produce misleading and false information when applied to non-aggregate queries, which is unacceptable in our setting. We have also experimentally shown that other AQP solutions (e.g., [3]), which create data subsets for later model training, do not provide accurate enough results for non-aggregate queries. The same holds for classical methods, including diverse sampling techniques and caching in managed database systems.

Proposed solution. We offer an innovative algorithm for efficiently approximating non-aggregate query results (full report, including the experiments is available at [2]). Given a database and expected query workload, our method creates a reduced data subset for fast approximate query processing when exact query runtimes are excessive. We define a metric for assessing the quality of a data subset, and show that it is computationally infeasible to find an exact optimized data subset. We therefore turn to Reinforcement Learning (RL) to find a good approximate solution. While advanced RL techniques are common in various domains, their application to tabular data management is limited due to several challenges, such as the large number of states and actions, lack of spatial or temporal dependencies, and absence of immediate feedback.

Our RL model, using critic-actor networks with Proximal Policy Optimization, is specifically designed for tabular data and draws inspiration from [3]. A pre-processing phase creates a reduced relaxed query workload ($C2$, $C4$), over whose query results a variational subsample is performed to reduce the initial data space ($C1$). A specialized reward function aligns with the goal of selecting rows for accurate approximations ($C3$). The approach yields an adaptive solution that significantly improves the efficiency and accuracy of non-aggregate query approximations. It also detects drift in user interests, enabling model fine-tuning, improving results ($C5$).

Benefits. Our solution, ASQP-RL, offers several benefits: (1) A versatile mediator adaptable to any database, enhancing flexibility across diverse environments. (2) Flexible training within time constraints, providing estimated guarantees and managing the tradeoff between running time and result quality. (3) Efficient query interpretation through vector-based techniques, enabling a rapid reduction of the initial space, facilitating swift training processes and contributing to overall system efficiency. (4) Incorporating a predictive model and query interpretation assesses the likelihood of missing answers, enhancing result accuracy. (5) The ability to detect interest drift, enabling fine-tuning to evolving exploration needs. This is achieved through the generation of initial query workloads and dynamic fine-tuning of the model with adapted queries. These benefits position ASQP-RL as a comprehensive tool for efficient and effective data exploration. Our approach can also handle aggregate queries in the query workload by rewriting them to select/project queries for the training phase. Aggregate queries can then be processed using the learned data subsets during data exploration.

Demonstration. Our demonstration highlights the ability of ASQP-RL to intelligently select tuples from real-world datasets for exploratory data analysis (EDA), as it seamlessly integrates with popular EDA platforms, providing users with a smooth and rapid exploration process. During the demonstration, participants will use Jupyter notebooks[1] for data exploration, and will see the speed with which query results are obtained as well as how similar the query results are to those obtained over the full dataset. The efficiency gains achieved by our system in terms of query response times are also showcased (further details are given in Section 3).

2 APPROACH

We now introduce the problem of Approximate Non-Aggregate Query Processing (ANAQP). We define the problem in a scenario with a known query workload, and give a metric to assess the quality of a set of subsets of tuples with respect to the given workload.

ANAQP Problem Definition. Consider a set of relational instances $\mathcal{T} = \{T_1, ..., T_n\}$ and query workload Q consisting of SPJ queries over $\mathcal{T}$. We denote w as a function $w : Q \to [0, 1]$ that assigns a weight to each query, satisfying the constraint $\sum_{q \in Q} w(q) = 1$.

We assume limits on the size of the available memory (k) as well as on the maximum result size that users can cognitively process (frame size F). In practice, F may vary from 10 rows (the default data view size in pandas) to 500 rows (the default result size in PostgreSQL), and is configurable.

Metric. Our metric function, $score(S)$, measures the quality of a set of subsets of the relational instances in $\mathcal{T}$, $\mathcal{S} = \{S_1, ..., S_n\}$ where $S_i \subseteq T_i$, with respect to $(\mathcal{T}, Q, w, k, F)$:

$$score(\mathcal{S}) = \frac{1}{|Q|} \sum_{q \in Q} w(q) \cdot min(1, \frac{|q(\mathcal{S})|}{min(F, |q(\mathcal{T})|)}) \qquad (1)$$

Intuitively, if a query returns more tuples than the frame size F, there is no need to include in S more than F tuples from the query result since the user may not be able to cognitively process them. Conversely, if a query returns only a few tuples, then each tuple carries significant importance in the result. The metric takes into account the weight assigned to each query and calculates the average score over all queries in Q.

ANAQP Goal. Given an instance $\mathcal{T}$, a set of queries Q, weight function w, and positive integers k and F, the goal of ANAQP is to

[1]Project Jupyter. https://jupyter.org

find a set $\mathcal{S}$ of subsets of tables in $\mathcal{T}$ where $\sum_{S_i \in \mathcal{S}} |S_i| < k$ such that $score(\mathcal{S})$ is maximized.

Parameter Settings. As can be seen from the problem formulation, the choice of parameter settings significantly impacts the overall performance of a solution. The frame size (F) directly affects the score function. The available memory (k) determines how large the subset can be and therefore affects the accuracy and coverage of query results. Increasing k enables a larger number of tuples to be kept, leading to a higher likelihood of obtaining exact query outputs and improved coverage for a wide range of queries.

Problem Hardness. ANAQP can be shown to be NP-hard by reduction to the max-k-vertex cover problem (proof omitted). Due to the problem hardness, we chose to use Reinforcement Learning (RL) to approximate a solution.

Unknown or Shifting Query Workloads. We have introduced *ANAQP* using a simplified scenario with a known query workload. However, the query workload is typically a subset of all possible user queries. Users continually issue new queries which may differ from the original workload, or the query workload may be unknown. We address this problem by generating queries that match user's interests when needed and defining the conditions necessary for re-training the model when an interest drift is identified (see [2]).

RL introduction. RL is a subfield of machine learning focused on developing algorithms and techniques in which agents learn optimal decision-making policies through interaction with an environment. The environment provides the agent with observations, and the agent's actions (choices made by the agent) can influence the environment. The feedback mechanism guiding the agent's learning process, reward signal, is a numerical signal that indicates the desirability or quality of the agent's actions. The current representation of the environment is referred as *state*, and it captures relevant information for decision-making and provides the context for the agent's actions. The strategy or set of rules that the agent follows to determine which action it should take when in a given state is referred as *policy*. During the learning phase, the agent may engage in exploration, taking unfamiliar actions to discover new knowledge, and exploitation, taking actions based on its current knowledge to maximize expected rewards. This is performed until the terminal state is reached, each iteration is referred as an *episode*.

Environment. In ASQP-RL we choose from a set of large tables the most relevant tuples with respect to a query workload, keeping within the available memory size. For this purpose, we defined an RL environment that learns gradually the approximation set (denoted *GSL*) as follows.

- *State*: The set of tuples chosen so far; This set is heterogeneous since it may contain tuples from different tables.
- *Initialization*: The initial phase is an empty set.
- *Action*: Adding a set of tuples from different tables.
- *Reward*: After performing an action a_t and arriving at a state S_{t+1}, we evaluate the score of the new state using the metric given in Equation 1 and referred to as $Score(S_{t+1})$. This is the reward the agent will get.
- *Episode*: An RL agent stops an episode when a terminal state is reached (when the agent reaches k tuples, k^{th} action).

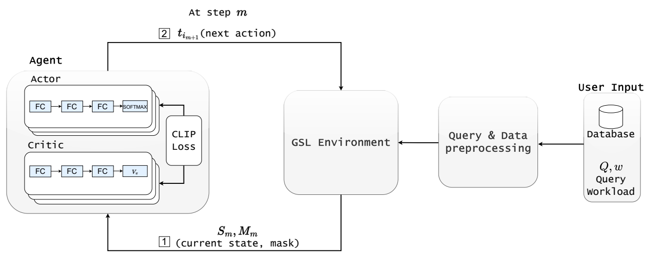
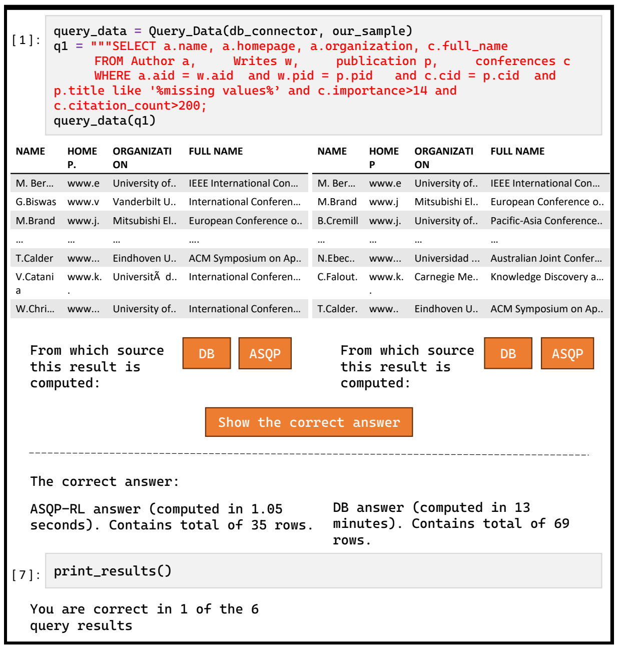


Figure 1: System Architecture

Our Approach. Our system architecture, ASQP-RL, is presented in Figure 1. For the agent, we used an actor-critic architecture with gradient clipping [10] (left-hand side of figure). Two networks, referred to as the actor and the critic, learn to choose actions and evaluate the chosen actions respectively. Gradient clipping is used to constrain the policy updates to be within a range of ε, preventing large updates that can cause instability. This choice of agent is backed by experiments that we have performed, as well as an ablation study with different environments and optimization levels (e.g. removing optimizations such as gradient clipping). For the environment (middle of figure), we use GSL since it allows the agent to explore freely and make independent choices for each tuple in the k-size subset. By combining the GSL environment with the aforementioned agent, our approach leverages the exploration freedom of GSL and the stability and efficiency of the agent, allowing our system to converge quickly and effectively optimize tuple selection.

Pre-processing. During the pre-processing phase (Figure 1, right), steps are taken to avoid overfitting to the known query workload and to reduce the search space. To mitigate the risk of overfitting to the query workload, we use query relaxation [5]. This introduces variations into the training data, encouraging the agents to generalize their learning beyond the specific queries encountered during training. Representatives of the generalized queries are then selected after transforming them to vectors, using a modification of sentence-BERT ($C2$). After executing the selected relaxed queries to determine the data subset that is used in some query, we further reduce the search space ($C1$) using variational subsampling to create an initial data subset, eventually transforming into vectors.

Further Improvements. Two additional improvements are made to use RL for our problem: (1) *Action Masking:* As discussed earlier, we reduce the size of the action space by only selecting tuples that appear in a relaxed version of the query workload. To avoid selecting the same tuple multiple times, we implemented action masking. Action masking enforces a constraint that forbids the agents from selecting certain actions, specifically in our case, those that would result in the repetition of a tuple. (2) *Diversity:* While not directly addressed by our approach, we found that the use of Reinforcement learning with its exploration policy leads to a diverse solution. We also tested adding a diversity component to our metric function, but found that it led to poorer results.



```
[1]: query_data = Query_Data(db_connector, our_sample)
     q1 = """SELECT a.name, a.homepage, a.organization, c.full_name
          FROM Author a,     Writes w,     publication p,     conferences c
          WHERE a.aid = w.aid  and w.pid = p.pid   and c.cid = p.cid  and
     p.title like '%missing values%' and c.importance>14 and
     c.citation_count>200;
     query_data(q1)
```

NAME	HOME P.	ORGANIZATION	FULL NAME
M. Ber...	www.e	University of..	IEEE International Con...
G.Biswas	www.v	Vanderbilt U..	International Conferen...
M.Brand	www.j.	Mitsubishi El..	European Conference o..
...	...	...	....
T.Calder	www...	Eindhoven U..	ACM Symposium on Ap..
V.Catania	www.k..	UniversitÃ d..	International Conferen...
W.Chri...	www...	University of..	International Conferen...

From which source this result is computed:

NAME	HOME P	ORGANIZATION	FULL NAME
M. Ber...	www.e	University of..	IEEE International Con...
M.Brand	www.j	Mitsubishi El..	European Conference o..
B.Cremill	www.j.	University of..	Pacific-Asia Conference..
...	...	...	...
N.Ebec..	www...	Universidad ...	Australian Joint Confer...
C.Falout.	www.k..	Carnegie Me..	Knowledge Discovery a...
T.Calder.	www..	Eindhoven U..	ACM Symposium on Ap..

From which source this result is computed:

```
The correct answer:

ASQP-RL answer (computed in 1.05        DB answer (computed in 13
seconds). Contains total of 35 rows.    minutes). Contains total of 69
                                        rows.

[7]: print_results()

You are correct in 1 of the 6
query results
```

Figure 2: Demo Presentation

3 SYSTEM AND DEMO OVERVIEW

The ASQP-RL system has been implemented in Python 3.9.13 and is available as an open source library[2]. It leverages Ray [9] for Reinforcement Learning, OpenAI's Gym [1] for the environment interface, and PyTorch, for the learning model of the agent. ASQP-RL can be integrated into common Exploratory Data Analysis (EDA) environments, such as Jupyter notebooks, and enables the efficient loading of subsets from any database.

To employ ASQP-RL, individuals need to execute the system's code in an offline setting. This process involves training the model, which begins by initializing it with both the database and the query workload. The output is a model with the ability to identify a subset of the database. The training phase may take from one to four hours long, depending on the user's parameter settings and the size of the data and workload. After this training, users have the option to use the model to generate a data subset. Generating the data subset occurs in real-time, and typically takes approximately three minutes to complete. Users can then explore this subset within an in-memory EDA environment. During the demonstration, a Jupyter notebook will be used for the EDA environment and the model will be pre-trained. Participants will see how executing queries in this environment is extremely efficient (just a fraction of a second) compared to querying the database directly. They will also see how close the answers are compared to querying the database directly.

Two scenarios will be used to show the benefits of ASQP-RL.

Demo Scenario 1: EDA. The first scenario uses the "Database" academic domain within the MAS database [8] to show how ASQP-RL can be used for EDA. The reason for selecting this domain is that users will have some intuition about interesting queries and the quality of query results. The session starts by introducing the MAS database schema, emphasizing tables, columns, and relationships. We will then start an EDA session in a Jupyter notebook, and show how the data subset can be generated from the pre-trained model and loaded into the notebook within three minutes. We then invite participants to step into the shoes of a data analyst by engaging them in a series of exploratory data analysis (EDA) queries, which are executed on the generated data subset. To provide a benchmark for query results, we also present pre-computed results from the complete database, ensuring delays elimination.

To engage participants further, the comparison will be done as a guessing game (depicted in Figure 2): Two results for the same query will be shown, and the participant must guess which result is over the full database and which is over the data subset. The query over the data subset is also compared with query over the full database in terms of the number of rows in the result and the query execution time. This demonstrates ASQP-RL's ability to produce answers comparable to the full database at a dramatically reduced execution time, and the participant's inability to discern the difference between the two. The session concludes by showing how many of the participant's guesses were correct (line [7] in Figure 2). To attract attendees further, we will maintain a scoreboard of users who have guessed the most accurately. As the participants are knowledgeable in this domain, they could write their own queries, and experience system's fast response time. Since querying the database directly is very slow, a direct comparison for user-defined queries will not be feasible during the demo.

Demo Scenario 2: Aggregation queries. Recall that aggregation queries in the query workload are turned into selection queries for training the model, but that aggregation can later be performed over the data subset. This scenario evaluates ASQP-RL's performance on aggregation queries for EDA. Aggregation queries are executed on both the database and ASQP-RL subsets. Comparisons between the database and ASQP-RL answers showcase its accuracy. For user-defined queries, ASQP-RL's subset-based responses are notably faster, highlighting its efficiency over database queries.

Acknowledgements We thank the BSF - the US-Israel Binational Science foundation grant 2018194, and iSF - the Israel Science foundation grant 2707/22 of the Breakthrough Research Grant (BRG) Program

REFERENCES

[1] Greg Brockman et al. 2016. Openai gym. *arXiv preprint arXiv:1606.01540* (2016).
[2] Susan B Davidson, Tova Milo, Kathy Razmadze, and Gal Zeevi. 2024. Learning Approximation Sets for Exploratory Queries. *arXiv preprint http://arxiv.org/abs/2401.17059* (2024).
[3] Park el al. 2018. Verdictdb: Universalizing approximate query processing. *Proceedings of the 2018 International Conference on Management of Data* (2018).
[4] Saravanan Thirumuruganathan et al. 2020. Approximate query processing for data exploration using deep generative models. In *ICDE*. IEEE, 1309–1320.
[5] Terry Gaasterland. 1997. Cooperative answering through controlled query relaxation. *IEEE Expert* 12, 5 (1997), 48–59.
[6] Viktor Leis et al. 2015. How good are query optimizers, really? *VLDB* 9, 3 (2015), 204–215.
[7] Kaiyu Li and Guoliang Li. 2018. Approximate query processing: What is new and where to go? A survey on approximate query processing. *Data Science and Engineering* 3 (2018), 379–397.
[8] Microsoft. 2016. *Microsoft academic search*. http://academic.research.microsoft.com
[9] Philipp Moritz et al. 2018. Ray: A distributed framework for emerging AI applications. In *OSDI*. 561–577.
[10] John Schulman et al. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[2]https://github.com/GalZeevi/SAQP