Towards Model-based Approximate Query Processing

Moritz Kulessa
Knowledge Engineering Group
TU Darmstadt

Benjamin Hilprecht
Data Management Lab
TU Darmstadt

Alejandro Molina
Machine Learning Lab
TU Darmstadt

Kristian Kersting
Machine Learning Lab
TU Darmstadt

Carsten Binnig
Data Management Lab
TU Darmstadt

ABSTRACT

In this paper, we present a new approach to Approximate Query Processing (AQP) called Model-based AQP that leverages deep generative models learned over a dataset to answer SQL queries at interactive speeds. Different from classical AQP approaches, deep generative models allow us not only to compute approximate responses to ad-hoc queries even over rare sub-populations but additionally support a new class of queries called counterfactual queries enabling users to ask what-if queries. Furthermore, we think that deep generative models can not only be used for AQP in databases but also have other applications for problems such as Query Optimization as well as Data Cleaning.

1. INTRODUCTION

Motivation. Interactive data exploration is an important tool allowing users to get an overview on new datasets. However, database systems do not offer interactive speeds for larger datasets. Hence, the database community has been working on different techniques for AQP which ensure fast query responses.

Unfortunately, existing AQP approaches suffer from various limitations that restrict the applicability to support the ad-hoc exploration of a new dataset [6]: (1) AQP approaches that are based on online sampling (e.g., DBO [8], CONTROL [7], approXimateDB [10]) are able to support ad-hoc queries but can only provide good approximations for queries over the mass of the distribution, while queries over rare sub-populations yield results with loose error bounds or even result in missing values in the query results. (2) AQP approaches that rely on offline sampling can use some form of biased sampling to mitigate this problem (e.g., AQUA [1], BlinkDB [2]), but therefore usually require a priori knowledge of the workload which is often not realistic if users want to explore a new database using ad-hoc queries.

1st International Workshop on Applied AI for Database Systems and Applications (AIDB'19), August 26, 2019, Los Angeles, California, CA, USA.

Contribution. In this paper, we propose a new approach for AQP called Model-based AQP that leverages deep generative models learned over the complete database to answer SQL queries at interactive speeds. Different from classical AQP approaches, deep generative models allow us not only to compute approximate responses to ad-hoc queries but additionally support counterfactual queries allowing users to ask what-if queries and validate hypotheses.

The idea of our approach is that deep generative models are either able to directly provide probability estimates that can be used to compute the results of simple count queries or to generate samples for more complex queries that could even include user-defined functions. In addition, we present an algorithm that supports the computation of expectations to allow average and sum aggregations.

In this work, we focus on pure analytical workloads where data is not updated online such as in data warehouses. However, recent methods for online learning of deep generative models [9] can be used to keep the model up to date as new data arrives. Furthermore, if the statistical properties of the data after the update do not change significantly, the approximate predictions would be similar. In this case, we can instead reuse the deep generative model and only need to update the meta-data (i.e., table sizes). Detecting this case efficiently is an interesting avenue for future work though.

We believe that deep generative models are also applicable to other problems in data management such query optimization or data cleaning. For instance, they could provide more accurate cardinality estimates for highly correlated data or they could be used for missing value imputation.

Outline. The remainder of this paper is organized as follows: In Section 2, we first give an overview of Model-based AQP and discuss the requirements a deep generative model has to fulfill to be used for AQP. We then show how SQL queries can be compiled into an Sum-Product-Network inference procedure using two different execution strategies based in Sections 3 and 4. Furthermore, we show that Model-based AQP is also able to answer counterfactual queries in Section 5. To validate the efficiency of our novel query processing approaches, we present our initial evaluation results using a real-world dataset in Section 6.

2. OVERVIEW

The main idea of Model-based AQP is shown in Figure 1. The deep generative model is built once over the original (potentially large) database and then used to answer SQL queries for data exploration in an interactive man-

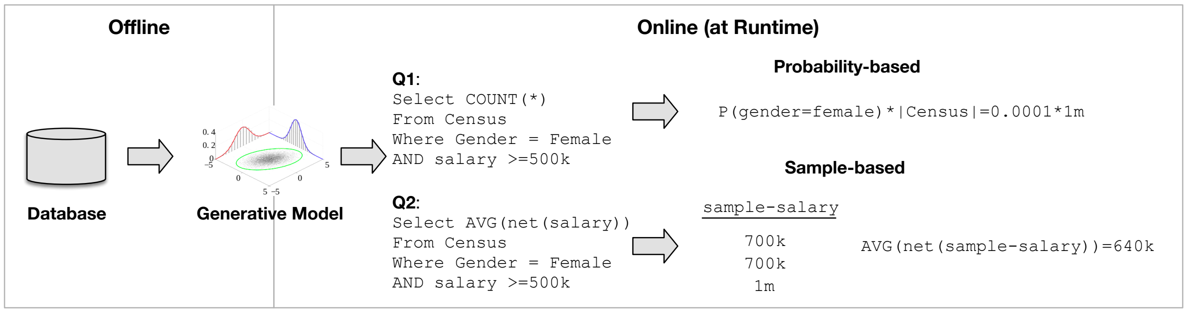

Figure 1: Overview of Model-based AQP.

Listing 1: Basic SQL-query with an aggregation.

```
SELECT G, AGGR(A) FROM T WHERE E GROUP BY G
```

ner. The general approach of Model-based AQP is thus similar to classical AQP creating a sample offline, which is used at runtime to answer queries. However, different from the sampling-based approaches Model-based AQP does not need to know the workload a priori (i.e., queries) to deal with rare sub-populations. In this paper, we support aggregate SQL queries with and without filter predicates as well as with and without group-by statements for Model-based AQP. Furthermore, we support user-defined functions and general arithmetic expressions to be used instead of base attributes. Joins and nested queries are not covered in this paper, but represent an interesting avenue for future work.

For approximate query processing, Model-based AQP provides different strategies as shown on the right-hand side of Figure 1: a probability-based strategy and a sample-based strategy. As shown on the right-hand side for query $Q1$, the probability-based strategy translates the given SQL query directly into an inference procedure and uses the resulting probability as well as statistics (i.e., the size of the table) to answer the query. The sample-based strategy instead is shown for query $Q2$. In this strategy the model is used to generate samples and then the samples are used to answer the query. Indeed, the probability-based strategy is more efficient than the sample-based strategy but can only be used for simple aggregate queries without user-defined functions or arithmetic expressions.

An essential requirement for Model-based AQP is that a deep generative model must enable tractable inference in hybrid domains, consisting of continuous and categorical distributions to support arbitrary database schemata. In particular, Mixed Sum-Product Networks (SPNs) [11] provide probability estimates and sample creation of even rare sub-populations and mixed domains with a complexity linear on the size of the model. We adapted the original SPNs with a modified leaf architecture and algorithms for the computation of expectations and counterfactual queries to make them suitable for AQP.

3. STRATEGY I: PROBABILITY-BASED

First, we want to introduce a query execution strategy that relies on expectation and probability estimations based on SPNs. This strategy is preferred since no samples have to be drawn and thus it is the most efficient strategy. However, it is only applicable to simple aggregate queries with no user-defined functions.

The main idea of this strategy is that a given SQL query as shown in Listing 1 can be parsed to obtain conditions corresponding to the filter predicate E and the group-by attributes G. Depending on the aggregation function, we have to perform different computations on the SPN. In the following, we discuss the details of the computation for the aggregation functions *COUNT*, *AVG* and *SUM*. *MIN* and *MAX* as well as other aggregation functions are currently not supported similar to other AQP approaches.

COUNT. To provide an answer for a query with the aggregation function *COUNT* we have to determine the number of entries of the queried sub-population defined by E. This can be estimated by multiplying the table size $|T|$ with the probability for the sub-population $P(E)$. Moreover, in case of a grouping we additionally have to analyze this sub-population with respect to the individual groups g defined by G yielding $P(E \wedge g) * |T|$.

Here, the probability $P(E \wedge g)$ represents that a record fulfills the filter condition E and is in the group g. In case E and g share conditions on same columns, these conditions are combined by taking the intersection of the set of possible values or value ranges. If the query does not specify a group-by statement the computation simplifies to $P(E) * |T|$.

AVG. For the computation of an *AVG* aggregation, we rely on the computation of expectations for the aggregation attribute A with SPNs. In case A is an arithmetic expression of columns, a result for the query can be computed if only the operators for addition and subtraction are used. With respect to these operators, we first compute the expectation for every single column of A individually which are then added or subtracted afterwards. In contrast, arithmetic expressions with multiplication and division operators cannot be computed with the probability-based approach. The computation of the approximation for each particular group $g \in G$ according to the sub-population E is then given by $E(A|E \wedge g)$

SUM. The computation of the result for a *SUM* aggregation can be reduced to the computation of a *COUNT* and an *AVG* aggregation by multiplying the respective results: $E(A|E \wedge g) * P(E \wedge g) * |T|$. For the same reason as for the *AVG* aggregation, the *SUM* aggregation can only be applied on single columns and on arithmetic expressions of columns which use the operators addition and subtraction. In any other case, no result with the probability-based approach can be computed and the sample-based approach, which is discussed next, will be used.

4. STRATEGY II: SAMPLE-BASED

In addition to the probability-based approach, the ability to generate samples with an SPN offers us another way to approximate the result for more complex aggregation queries which use user-defined functions or arithmetic expressions. Contrary to classical sample-based approaches for AQP, we can use the SPN to produce samples at query time without even accessing the real data. In particular, we are able to generate biased samples, which is one of the most significant advantages. In this work we propose three different sampling techniques which are explained in the following sections.

Random Sampling. First, as a baseline we introduce the generation of random samples with the SPN for which we use the sample functionality of the SPN without specifying any conditions. Like for classical random sampling from data, we face the issue that samples can be generated which are not relevant for answering the submitted query. In particular, if the SQL query is only applied on a small sub-population of the data many generated samples are discarded.

By using random samples, the query result can be approximated. In case of an AVG aggregation, no modifications have to be made since the result of an AVG aggregation is independent of the number of entries on which it is computed. In contrast, $COUNT$ and SUM aggregations depend on the number of samples on which they are computed. Therefore, we have to scale-up the result of these aggregations to get the approximation. This is done by multiplying the result with the total number of entries of the data $|T|$ divided by the number of the samples $|S|$ which have been generated: $m_{random} = \frac{|T|}{|S|}$.

Relevance Sampling. In order to avoid the generation of irrelevant samples, we use another more advanced approach called relevance sampling. This approach only generates samples for the queried sub-population defined by the filter predicate E. Compared to the random sampling approach, we can improve the efficiency of approximating the result, especially for rare sub-populations, since we do not have to discard any samples. For instance, if only 1% of the data is relevant to answer the SQL query, then the relevance sampling approach is one hundred times more efficient than the random sampling approach to obtain the same precision for the approximation.

Due to the sample generation, the approximation of the result for the aggregation is different compared to random sampling. To scale-up the result of $COUNT$ and SUM aggregation queries, we further need to multiply the result with the probability $P(E)$ of the sub-population of the data. This probability is obtained by performing inference for the sub-population specified by the filter E. The respective multiplier used to scale-up the result is $m_{relevance} = \frac{|T|}{|S|}P(E)$.

Stratified Sampling. The relevance sampling approach is already a major improvement for the approximation of aggregation results compared to random sampling but it ignores the grouping of the SQL query. Each specific queried group should obtain an approximation as fast as possible. However, the relevance method does not consider the selectivity of the queried groups which, in fact, can be skewed. Hence, particular groups will obtain more samples to approximate the result than other groups which has an effect on the precision of the approximations.

Listing 2: Counterfactual query.

```
SELECT ... GROUP BY G WITH C TO X
```

This problem can be solved with the stratified sampling approach which can generate samples for each particular group g independently. For rare groups we restrict the number of samples to avoid an over-representation. In particular, we only generate as many samples for a particular group as tuples of that group are available in the original data. This information can be obtained by performing inference on the SPN with the conditions for that particular group. The remaining number of samples, are distributed evenly over the other groups.

Similar to relevance sampling, we have to adapt the computation of the approximation, because we rely on biased sampling. Since we are generating the samples for each group independently, the aggregation result has to be approximated for each group on its own. Similar to the other proposed sampling approaches, the multiplier only needs to be applied to scale-up the result of the $COUNT$ and SUM aggregations: $m_{stratified}(g) = \frac{|T|}{|S(g)|}P(E \wedge g)$, where $|S(g)|$ represents the number of samples of a group g.

5. COUNTERFACTUAL QUERIES

Often in interactive data exploration, the question arises what the impact of a slightly different distribution of the population would be. For example a clothing company might wonder how the annual sales would change, if the number of high income customers was 50 percent higher. This can have an impact on future decision making of the company such as addressing a specific customer group with the advertisement. As stated in [3], a counterfactual sentence has the form of 'If A was true, then C would have been true' where A specifies an event that is contrary to one's real world observations and C specifies the result that is expected to hold in the alternative world where A is true. This idea can be transferred to aggregation queries such as the ones of Listing 1 as well. In this case we are interested in the change of the aggregate given that a certain sub-population is increased or decreased.

For the representation of counterfactual queries we introduce a new SQL extension as shown in Listing 2. Here, C represents the combination of conjunctive conditions defining the sub-population and X defines the scaling factor. To support this class of queries, we developed an algorithm that allows to answer such queries efficiently with SPNs. Before our AQP techniques are executed, the SPN is adapted to the changed population defined by the counterfactual condition.

6. INITIAL EXPERIMENTAL RESULTS

As a major aspect of our initial experimental evaluation, we evaluate our proposed AQP techniques on a real-world dataset containing U.S. domestic flights [4] and used the data generator of [5] to scale it to $10M$ instances. As baselines we compare our model-based AQP approaches to random sampling from data and exact SQL. We did not compare to approaches that rely on offline sampling such as [2] since these approaches require that the workload is known in advance; i.e., they do not support ad-hoc queries.

In order to evaluate our approaches, we provide a variety of different queries where we not only varied the aggregation

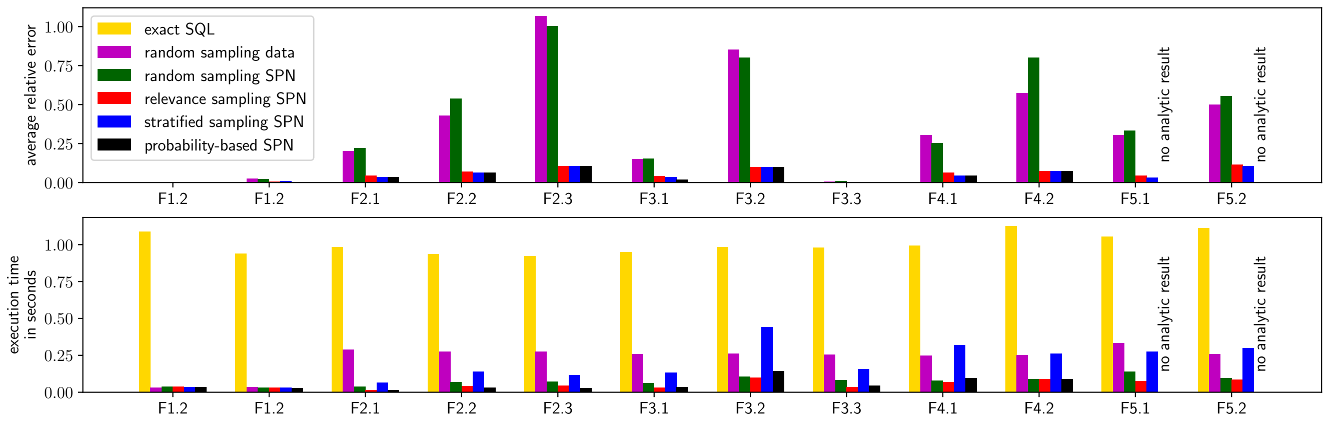

Figure 2: Results of the average relative error and the execution time for all queries.

function but also the number of group-by values and the selectivity (sel) as shown in the following table as queries (*F1.1* to *F5.2*). The query set includes also some complex queries with UDFs (*F5.1* and *F5.2*) where our probability-based approaches could not be applied and thus we resorted to sampling from the model:

identifier	aggregation	groups	sel in %	skewness
F 1.1	AVG	1	$\sim$ 5.6000	-
F 1.2	AVG	1	$\sim$ 1.3800	-
F 2.1	COUNT	26	$\sim$ 1.0000	1.4343
F 2.2	COUNT	26	$\sim$ 0.1260	1.4122
F 2.3	COUNT	22	$\sim$ 0.0140	0.4719
F 3.1	SUM	22	$\sim$ 1.0000	0.0329
F 3.2	SUM	53	$\sim$ 0.1200	2.4420
F 3.3	AVG	26	$\sim$ 0.1500	1.4735
F 4.1	COUNT	53	$\sim$ 1.3600	2.4234
F 4.2	COUNT	26	$\sim$ 0.1000	1.5480
F 5.1	SUM	22	$\sim$ 0.1402	0.2504
F 5.2	SUM	53	$\sim$ 0.5213	2.4838

The skewness of the group-by column Y is computed as:

$$skewness(Y) = \frac{\sum_{i=1}^{|Y|}(Y_i - \bar{Y})^3/|Y|}{std(Y)^3} \quad (1)$$

The results for the average relative error and the execution time for all evaluated queries are shown in Figure 2. The main observation is that our proposed approaches are able to process all our queries with typically less computation time compared to exact SQL and better accuracy compared to random sampling. In particular, for the sample-based approaches, we report the execution time of the queries until the average relative error is below 5% and full bin completeness is achieved. Therefore, we generate samples until the specified goal or the limit of $100,000$ instances is reached. In case that the limit is reached, we stop the sampling procedure and report the error of the results which is achieved with the respective number of samples. Since exact SQL and the probability-based approach do not rely on samples, we only report the average relative error and the execution time for these approaches.

In particular, if we investigate the execution time of the random sampling approaches we can see that the SPN is much more efficient in the sample creation while the quality of the approximations is similar. This is due to the fact that we do not have to deal with the entire dataset during query time when using the SPN. Further improvement in efficiency and therefore also in quality of the approximations can be achieved by using the relevance sampling approach. Moreover, we can observe a slight increase of the execution time for stratified sampling and the probability-based approach on queries with a high number of groups (e.g. query *F3.2*). The reason for this is that each group needs to be handled individually by these approaches. Furthermore, our stratified sampling approach outperforms the other sampling approaches in quality when the skeweness is high (e.g. queries *F4.1* and *F5.2*).

7. REFERENCES

[1] S. Acharya et al. The Aqua Approximate Query Answering System. In *Proceedings of the 1999 ACM SIGMOD International Conference on Management of Data*, pages 574–576, 1999.

[2] S. Agarwal et al. BlinkDB: Queries with Bounded Errors and Bounded Response Times on Very Large Data. In *Proceedings of the 8th ACM European Conference on Computer Systems*, pages 29–42, 2013.

[3] A. Balke et al. Probabilistic Evaluation of Counterfactual Queries. In *Proceedings of the Twelfth AAAI National Conference on Artificial Intelligence*, AAAI'94, pages 230–237. AAAI Press, 1994.

[4] Bureau of transportation statistics. `http://www.transtats.bts.gov`, 2017. Accessed: 2019-05-03.

[5] P. Eichmann et al. IDEBench: A Benchmark for Interactive Data Exploration. In *Proceedings of the Very Large Data Base Endowment*, volume 11, 2018.

[6] A. Galakatos et al. Revisiting Reuse for Approximate Query Processing. *Proceedings of the Very Large Data Bases Endowment*, 10(10):1142–1153, June 2017.

[7] J. M. Hellerstein et al. Interactive Data Analysis: the Control Project. *Computer*, 32(8):51–59, August 1999.

[8] C. Jermaine et al. Scalable Approximate Query Processing with the DBO Engine. *ACM Transactions on Database Systems*, 33(4):23:1–23:54, December 2008.

[9] A. Kalra et al. Online structure learning for feed-forward and recurrent sum-product networks. In *Advances in Neural Information Processing Systems*, pages 6944–6954, 2018.

[10] F. Li et al. Wander Join: Online Aggregation via Random Walks. In *Proceedings of the 2016 International Conference on Management of Data*, pages 615–629, 2016.

[11] A. Molina et al. Mixed Sum-Product Networks: A Deep Architecture for Hybrid Domains. In *AAAI*, 2018.