Towards a benefit-based optimizer for Interactive Data Analysis

Patrick Marcel, Nicolas Labroche
University of Tours
Tours, France
firstname.lastname@univ-tours.fr

Panos Vassiliadis
University of Ioannina
Ioannina, Greece
pvassil@cs.uoi.gr

ABSTRACT

This vision paper introduces several ideas around the optimization of Interactive Data Analysis (IDA) tasks. With an eye on traditional query optimization (QO) in Relational DataBase Management Systems (RDBMS), we suggest that IDA tasks should be specified through high-level statements and optimized globally, particularly by maximizing the number and significance of insights that can be automatically collected for the task. We envision an architecture for IDA and propose in the context of IDA what corresponds to statistics, cost model and execution plan pruning strategy in relational systems. We give elements pertaining to the feasibility of the vision and draw perspectives.

1 INTRODUCTION

Interactive Data Analysis (IDA) corresponds to the process of exploring a dataset by means of a sequence of actions aiming at answering an often imprecise user information need [13, 18]. Let us quote [18]: *Data analysis is fundamentally an interactive, iterative process in which a user issues an analysis action (i.e. query), receives a results set, and decides if and which action to issue next. Until recent years, analysis tasks required thorough expertise in SQL and programming, as well as mathematics and statistics. However, since the advent of the Big Data era, the infrastructures and support for Interactive Data Analysis (IDA) [...] are gradually replacing traditional tools, allowing easy to-use data exploration, visualization, and mining, even for users lacking knowledge of SQL and programming languages. Yet, IDA is still a difficult process, especially for inexperienced users, as it requires a deep understanding of the investigated domain and the particular context. Users may therefore skip significant analysis actions and overlook important aspects of the data.*

Interestingly, it is already envisioned a fully automated IDA process called continuous exploration [28]. It is therefore important to be able to optimize this process and its steps.

Topic of this paper. This paper's topic is the discussion of an envisioned query optimization process, that goes beyond traditional query processing and optimizations and addresses the issue of efficiently computing results to interactive data exploration steps, with these results going far beyond the simple delivery of tuples (as nowadays happens), but including tuples, data mining results on these tuples, as well as text and visualization generation.

Traditional optimization. If a RDBMS is used in IDA, i.e., the action corresponds to the execution of an SQL statement, the full power of RDBMS query evaluation is unleashed to optimize one action of the IDA task, which can be seen as a local optimization. In these systems, query optimization is typically done at runtime, and corresponds to the selection of a physical query execution plan whose cost is estimated to be the best among alternative execution plans. The cost of a plan is estimated based on statistics collected over the database objects (tables, indexes, views, etc.), and as the search space of physical execution plans is huge, pruning strategies are used to ensure the query optimization phase of the query evaluation process is done in a reasonable time. The computation of the cost of a plan depends on a specific cost model used by the DBMS, that usually depends on the number of blocks (data, index, etc.) needed to process the query [8].

Alternatively, if the action is the use of a ML algorithm, the algorithm itself includes its own optimizations (for instance, an FP-growth implementation can be used instead of a level-wise implementation for association rules extraction).

Vision. In what follows, we consider explorations where a user interactively queries some data sources and apply Machine Learning[1] (ML) algorithms to extract models or patterns out of the query answers, to find valuable insights in the data. We foresee that IDA users will express such explorations with a high-level declarative language. But differently from RDBMS, and given the explorative nature of IDA, *we assume that such high-level statements will not be prescriptive, in the sense that the atomic actions (queries over DB, application of ML algorithms) and their combination will be left to the statement processor.* We believe, though, that a declarative language remains needed, and, for instance, simple statements like keyword search may not be suitable for expressing complex IDA tasks. If traditional query optimization tackles the problem of optimizing one particular step of IDE (i.e., relational query processing), *there is a need to optimize the whole sequence of steps, including querying data, extracting models, etc.*

We are quite emphatic in our vision, that the current notion of query result needs to be fundamentally questioned. In our vision, the traditional treatment of queries, holding sets-of-tuples as results, need to be replaced by data story answering. A "data story" is the answer to an intentional, high-level query via the composition of individual results that (a) address the core of the original query, (b) contextualize it, by comparing the state of the situation that the query result describes with similar, relevant contexts, (c) analyze and explain it, by highlighting critical sub-parts of the data space that are responsible for the observed state, and (d) summarize key highlights of this mini-exploration into a concise summary. To achieve this, apart from asking database queries, it is imperative that the engine(s) involved complement the retrieved data with the mining of ML models from them and automatically generate textual descriptions and visualizations in a coherent sequence of "data episodes" (dashboards with data, ML results, text and visuals) that compose the data story.

Example. We showcase our vision with an example inspired from [26]. Assume a user is analyzing sale results in a dashboard

[1] We use Machine Learning as a general term to denote supervised/unsupervised descriptive/predictive methods.

narrating the current state of sales through a collection of cross-tabs and graphics. The user then would like to: "*verify* whether this distribution of sales for mfgr#5 in Argentina from 2011 to 2016 *still holds in general*, and build a clustering model for it, then backtrack to compare with sibling countries, and finally *explain* the highest country-wise difference." This request would correspond to the following statement in the intentional language:

$$explain_{highlight:MaxDifference}($$
$$\quad compare_{siblingCountries}($$
$$\quad\quad backtrack($$
$$\quad\quad\quad cluster($$
$$\quad\quad\quad\quad verify_{mfgr\#5,Argentina,2011-2016}(\text{current dashboard})))))$$

This statement is non prescriptive in that it is left to the optimizer to decide the best roll-up for the verification, the best algorithm and number of clusters for the clustering, the best way to explain the difference, etc. Each of these degrees of freedom will give rise to a new plan, yielding an answer different from those of the other plans, automatically generated with the relevant data sources identified from the current dashboard. The optimizer estimates the benefit of each plan in terms of the number of insights, their significance, the time to extract them, etc., and eventually translates the best plan into a sequence of operations, for instance: roll-up, cluster, Cinecube's put in context [10], and Diff [20]. These operations are evaluated and the insights (e.g., major deviations in the distribution of sales, prototypes of the clusters, major contributions to the highest country-wise difference) in each answer are highlighted. A data story is then produced by sequencing the answers into a new dashboard and choosing for them the best graphical displays.

Contributions. The contribution is a vision for a global (as opposed to local) optimization scheme for IDA, where:

- the user issues a declarative *intentional statement*, expressing her high-level data analysis goal in a non-prescriptive way, i.e., without completely specifying the data sources and operations to apply over them,
- this statement sketches a complete exploration, or *data story specification*, i.e., a sequence of complex actions including data retrieval, model extraction with ML, automatic selection of key insights, and visualization. These atomic actions derived from the statement (appearing in the expression tree used for the execution plan) are DB queries, ML algorithms, etc., expressed in terms of *atomic operators* of the underlying execution engine,
- an *optimizer*, relying on a specific *cost model*, that is essentially related to the number, extraction cost, properties and significance of insights gained by the user along the exploration, decides how to combine the operators into alternative plans and picks the best one on the basis of a *strategy* defined for pruning the search space of execution plans.
- the involved execution engine(s) produce intermediate results that are combined, ranked, pruned with respect to their significance and relevance to the user's goal and eventually compiled into a *data story* that is shown to user in response to her query statement.

This vision paper is structured as follows. Section 2 discusses cost-based optimization and insights. Section 3 exposes the envisioned architecture. Section 4 presents the optimizer while Section 5 focuses on its cost model. Finally, Section 6 discusses a pruning strategy and Section 7 draws perspectives.

2 COST-BASED OPTIMIZATION AND INSIGHTS

We now motivate the need for a cost-based optimizer, that relies on a cost model where insights are first-class citizens.

Why a cost-based optimizer? Merging DB with ML is not new. A recent Sigmod blog post reports the interview of DB and ML academics and industrials discussing this matter [1]. In particular, model selection is mentioned in the discussion as an important challenge. But mixing ML actions with DB queries (for instance, computing a set of decision trees over the data of a query, or, even more far-reaching, selecting the most informative out of the ones that are generated to this end) is a complex problem. We can distinguish two basic options to do so: (i) use a rule-based approach: propose a set of fixed rules to derive queries and applications of ML algorithms, based on the semantics of the high-level statement. The rules are applied in a specific order depending on the statement. Given a statement, the derivation corresponds to the execution plan. (ii) use a cost-based approach: one intentional statement generates several execution plans with different operator instantiations. The cost of each plan is estimated and the best one is retained. The cost model is defined by means of an objective function that encodes the intuition of what the optimal solution should be.

At some point of the discussion of the blog's post, one of the interviewed insists on the importance of cost-based schemes compared to the more ad-hoc rule-based ones. Interestingly this kind of transition from rule-based optimization to cost-based optimization is what had made DB successful in terms of query optimization.

Why insights in the cost model? Insight is a fundamental driver of the IDA process. In [6], a benchmark for IDA is outlined. The authors insist on the importance of insights. Quoting this paper: *Ultimately, the goal of interactive data exploration is to extract insights from data. Thus, a system that allows to extract more insights than another system within a given time frame is preferable. However, creating a measure that captures this notion in a comparable and reproducible way is hard. What is an insight? Do different users have different notions of insights? How do we measure the complexity and value of an insight? Are they domain-dependent?*

In [6], the authors argue that directly measuring insights per minute cannot be done efficiently and they propose a proxy metric that reflects how often and by how much a system violates a latency requirement (for instance the interactive response time).

We agree that insights, their number, the time it takes to discover them, should be of primary importance when processing IDA tasks. At the same time, we also believe that other properties, like for instance diversity of insights, significance of insights, etc., should be taken into account.

Yet, what does insight *mean?* Insights are human-digestible pieces of interesting information about the data [3]. Let us quote [10]: *In a recent approach, Dove and Jones [5] combine the definitions from the communities of Information Visualization and Cognitive Psychology: whereas the InfoVis community defines insight as "*something that is gained*" (after the observation of data by a participant), psychologists define it as an* "Aha!" moment which is experienced. *Interestingly, the two definitions can be combined in a common view, where once the user works with information, starting with an original state of mind on the current state of affairs, there is an "Aha!" moment, where the user suddenly realizes a new*

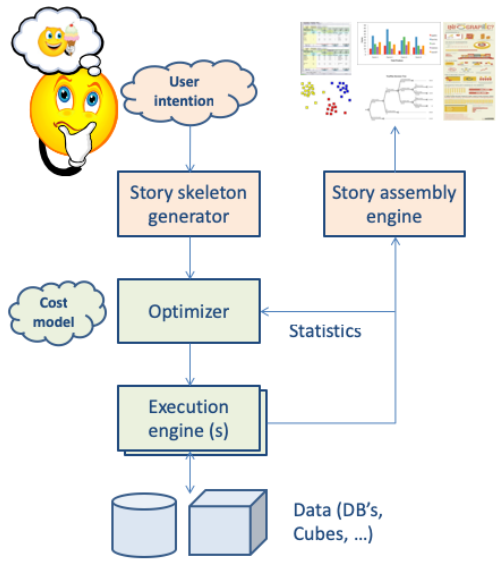

Figure 1: The envisioned architecture

way of looking at the data, resulting in a new mental model for the state of affairs, or else, new understanding [5].

To facilitate the "Aha!" moment, several works already proposed to characterize unexpected values in cubes ([21], among others), interesting [9] or unexpected [3] patterns in data, statistically significant relationships between data sets [4] and hidden causes [20]. We note that control mechanisms can be included [30] to enforce insight validity.

3 ENVISIONED ARCHITECTURE FOR IDA

We assume the following multi-tier architecture for intentional IDA:

- a user layer: in this layer, the user expresses intentional statements asking for data stories;
- a story skeleton layer: in this layer, the translation of the formal intentional statement(s) to a data story skeleton (i.e., structure of requested operations, without the results that will have to be plugged in) takes place;
- an optimizer layer: this layer is responsible for the translation of the intentional statement into atomic actions picked from a catalog, that are automatically tune and parametrized;
- an execution layer: that executes queries, applies models, extracts highlights, etc.;
- a data layer: this layer holds databases, cubes, flat files, user profiles, user traces, etc.;
- finally, a storytelling layer: this layer is responsible for compiling all the intermediate results into meaningful data stories.

More details about the layers. At the user layer, statements can be intentions in the spirit of those defined in [26, 27], or translated into such language using NLP facilities. This layer is also responsible for long-term and sort-term modeling of the user [11, 29], by processing implicit user feedback (e.g., intention logs). While in traditional RDBMS a query computes an instance from another instance, an intentional statement expresses a data story starting from another data story. A data story is a sequence of visualizations packed in a dashboard, produced by the storytelling layer from the output of the optimization and execution layers. Predefined user templates will be used to control the story complexity, and to generate the initial data story, to start the analysis. The storytelling layer identifies the most appropriate graphical representation of query answers, and automatically crafts narratives, commenting on the highlights presented, etc. [10, 12]. As in web search SERP (Search Engine Result Page), a set of stories resulting from the processing of an intentional statement can be presented in a way that allows the user to navigate the most relevant data stories. We note that explanations of the models extracted from the data [19] as well as user-tailored recommendations [18] may also be part of the story. The execution layer is responsible for choosing the most efficient way to execute the atomic operation (i.e., it applies local optimizations). We now detail the optimization layer.

4 OPTIMIZER

The optimizer layer processes an intentional statement, produces several query plans, evaluate their cost and picks the best one. A query plan is a tree where nodes are atomic operations, i.e., either regular queries over data sources or calls to ML algorithms, in the spirit of the trees used in [18]. This supposes to have a catalog of such operators. An objective function is used to estimate, for each node in the tree, the cost or benefit of the nodes, in the sense of some objective function to optimize (See Section 5).

One major difference compared to the classical query optimization scheme is that, given an intentional statement, two different query plans for this statement generally result in two different answers, whereas in traditional QO, different execution plans correspond to a query that is logically equivalent to the query being optimized. Another fundamental difference is that in classical QO, the user sees only a set of tuples as the final result. Here, insights encountered along the plan are collected and post-processed for telling the data story, along the lines delineated in the vision presented in the introduction of the paper.

How to automatically generate a set of relevant queries to a database? This should obviously be partly specified by the intention. Many works exist to generate consistent queries from incomplete specifications [23].

How to automatically choose and tune ML algorithms? Choosing a learning algorithm based on data characteristics is the topic of a field of research called meta-learning [16, 25]. Quoting [16]: *A considerable amount of metalearning research has been devoted to the area of algorithm recommendation. In this special case of metalearning, the aspect of interest is the relationship between data characteristics and algorithm performance, with the final goal of predicting an algorithm or a set of algorithms suitable for a specific problem under study. As a motivation, the fact that it is infeasible to examine all possible alternatives of algorithms in a trial and error procedure is often given along with the experts necessary if pre-selection of algorithms is to take place. This application of metalearning can thus be both useful for providing a recommendation to an end-user or automatically selecting or weighting algorithms that are most promising.*

The basic principle of metalearning is as follows: given a set of datasets, extract characteristics of the datasets, estimate performance of the algorithms on each dataset (obviously with an indicator that allows to compare different algorithms), build a meta-dataset describing for each dataset its characteristics and

the estimated performances, and use it to rank the algorithms. In [14], the model selection is viewed under the angle of manipulation of triples. Each triple has three components: an FE (feature extraction) "option" (loosely defined, a sequence of computation operations) that fixes the feature set that represents the data, an AS (algorithm selection) option that fixes the ML algorithm, and a PT (parameter tuning) option that fixes the parameter choices conditioned on the AS option.

Automated machine learning (auto-ML) [7] is very similar to meta-learning in its purpose. It focuses on how to choose and parametrize a ML algorithm for a given dataset, at a given cost (i.e., the time it takes to test different algorithms).

5 COST MODEL

Traditional query optimizers are usually concerned with minimizing resource consumption, especially time or disk access, and they use a cost model in accordance. In the context of IDA, we envision the definition of an objective function to express the gain the user would get from the exploration. As explained above, and consistently with the literature on IDA, this benefit function should attach a particular importance to the insights and their significance, without forgetting the traditional concern of resource consumption (in particular, local optimizers will be exploited as black boxes attached to the data sources used during IDA). Here is a non-exhaustive list of criteria:

- the number of insights (to maximize; albeit, not to infinity, but to a concise set of top-k results),
- the time it takes to obtain them (i.e., processing the intention) (to minimize),
- some properties of insights or sets of insights:
  - their statistical significance (minimize p-value, see e.g., [30]) (to maximize),
  - their relevance for the user (e.g., with respect to some user preferences) (to maximize),
  - their interestingness [9] (to maximize): understandability (e.g., the size/length of the insight or the complexity of a model or pattern), diversity, etc.
- the appropriateness of the insight to the current intention (e.g., in the sense of a short-term interest) (to maximize).

Some more criteria related to the way data and insights are displayed in the data story may also be included.

As in classical QO, we envision a mechanism of statistics collection to estimate the benefit of the exploration with the objective function. For instance, the average processing time of ML algorithms can be recorded, to prevent a costly algorithm from being considered in plan generation whenever this algorithm is to be applied over a dataset having similar characteristics than those used in the past. Since global query plans are expected to be complex, we anticipate that global optimization will be a good candidate for query re-optimization [17] or plan reuse [22], in order to reduce the overhead induced by the optimization phase.

The objective function is used to seek a compromise between the above-mentioned criteria, since some are antagonistic. We note that it can be automatically learned and maintained using supervised ML. For instance, in [31], the authors propose to use active learning to predict a set of Pareto-optimal solutions with some theoretical guarantees in a multi-objectives optimization context. More simply, it is possible to use a scalarization approach that reduces the multi-objective optimization problem to a single-objective problem. A naive solution could be a linear combination of the criteria to be learned using SVM, provided user feedback and enough instances of intention processing are available, using an approach similar to the one presented in [11].

6 PLAN GENERATION AND PRUNING STRATEGY

The search space for plan generation is huge, much more than the one used by traditional QO. A drastic pruning strategy is needed, to reduce the number of plans generated. We here present a naive idea to show the feasibility of plan generation and selection, that borrows from composite item construction strategy (see e.g.. [2]).

Given an initial data story, we form combinations of datasets and algorithms, and order them to tell the story. The algorithm below is a first tentative to formalize the process of picking an execution plan as the answer to an intentional operator.

Algorithm 1: main

Data: datastory D, intention I, benefit function f_o, integer n
Result: set of execution plans P

```
1 A = generateNodes(D, I); // generate a set of plan nodes N
    from D and I
2 C = buildCandidateClusters(n, N); // cluster N based on
    criteria
3 C' = prune(C, f_o) ; //prune C
4 for all cluster c in C' do
5 |   p_c = queryPlan(c) ; // compute query plan
6 |   estimateBenefit(p_c, f_o) // estimate its benefit
7 |   P = P ∪ p_c
8 return P;
```

The algorithm works as follows. In step 1, the intentional operator triggers the generation of a large set of execution plan nodes, i.e., atomic operations. Clustering is used in step 2 to build n clusters of nodes, n being a parameter of the approach. Step 3 kills some clusters with hard constraints. Each remaining cluster is processed to obtain a query plan, and all query plans are scored using the benefit function (steps 4-6). Finally, the set of scored query plans is returned.

We now give some details on the different steps. (1) generateNodes: the generated atomic operations are queries over datasources or ML algorithms. Queries over sources are generated using the intention and user preferences, while ML algorithms are taken from a catalog. Given the size i of intention (in terms of number of sources), the size p of user preferences (for instance, selection conditions to expand the query with [24]) and the number m of ML algorithms, the number of generated operations is around $i \times 2^p \times 2^m$. (2) buildCandidateClusters: all atomic operations are described in a uniform way. They are projected into a predefined vector space (including dataset characteristics, ML algorithm characteristics, etc.), so that clustering can be used to form groups of operation being "close" to each other. Like in [2], fuzzy clustering can be used, as it allows for an atomic operation to appear significantly into several execution plans, with extra terms to incorporate the criteria of the objective function. In this step, it is crucial to ensure that the clustering time will remain under control. For instance, avoiding necessary comparisons based on statistical notions like concentration inequalities [15] could be used. (3) prune: this step uses a subset of the criteria of the benefit function as hard constraints for pruning clusters that are considered too bad (for instance, the worst clusters in terms of overall execution time of the operators they contain can be

automatically rejected). (4) queryPlan: this step orders the nodes of the cluster, and adapts ML algorithms to query answers thanks to auto-learning [7]. (5) estimateBenefit: consists of applying the benefit function to each node of the plan.

7 CONCLUSION

This paper introduces a vision for processing high level statements describing Interactive Data Analysis tasks, inspired by traditional query optimization in relational databases. To the best of our knowledge, this is the first attempt at an insight-driven architecture for analytical intentions. It is consistent with previous works on IDA: Eichmann and al. [6] that discusses the challenges for creating an insight-centered benchmark for IDA, where different functional aspects of IDA are separated ; Zhao et al. [30] focusing on the validity of a certain type of insights encountered along the analysis ; or Milo and Somech [18], that proposes a collaborative, but insight-agnostic, recommender system for IDA where analyses over diverse datasets are phrased with high-level actions of multiple types.

We believe that our vision opens many research directions, that can be structured around the envisioned architecture and its layers. Specifically, each aspect of the optimizer deserves to be refined, i.e., (i) the categorization of insights, and a precise definition for them, (ii) the objective function, its criteria and the way is it learned, (iii) the mechanism for statistics collection and user feedback, (iv) the vector space for representing atomic operations in a uniform way, and (v) the different steps of the pruning strategy.

Given the maturity of the different fields involved in this vision (databases, machine learning, user-centered search activities, etc.), we are confident that a proof of concept can be implemented once these different aspects are precised. We note that the dataset provided by [18] and made available through Github[2] is a promising option for carrying a set of tests with real IDA tasks.

REFERENCES

[1] A. Abouzied and P. Papotti. Courting ML: Witnessing the marriage of relational & web data systems to machine learning. http://wp.sigmod.org/?p=2243, 2018.

[2] M. Alsayasneh, S. Amer-Yahia, É. Gaussier, V. Leroy, J. Pilourdault, R. M. Borromeo, M. Toyama, and J. Renders. Personalized and diverse task composition in crowdsourcing. *IEEE Trans. Knowl. Data Eng.*, 30(1):128–141, 2018.

[3] T. D. Bie. Subjective interestingness in exploratory data mining. In *IDA*, pages 19–31, 2013.

[4] F. Chirigati, H. Doraiswamy, T. Damoulas, and J. Freire. Data polygamy: The many-many relationships among urban spatio-temporal data sets. In *SIGMOD*, pages 1011–1025. ACM, 2016.

[5] G. Dove and S. Jones. Narrative visualization: Sharing insights into complex data. In *IHCI*, 2012. Available at http://openaccess.city.ac.uk/1134/.

[6] P. Eichmann, E. Zgraggen, Z. Zhao, C. Binnig, and T. Kraska. Towards a benchmark for interactive data exploration. *IEEE Data Eng. Bull.*, 39(4):50–61, 2016.

[7] M. Feurer, A. Klein, K. Eggensperger, J. T. Springenberg, M. Blum, and F. Hutter. Efficient and robust automated machine learning. In *NIPS*, pages 2962–2970, 2015.

[8] H. Garcia-Molina, J. D. Ullman, and J. Widom. *Database systems - the complete book (2. ed.)*. Pearson Education, 2009.

[9] L. Geng and H. J. Hamilton. Interestingness measures for data mining: A survey. *ACM Comput. Surv.*, 38(3):9, 2006.

[10] D. Gkesoulis, P. Vassiliadis, and P. Manousis. Cinecubes: Aiding data workers gain insights from OLAP queries. *Inf. Syst.*, 53:60–86, 2015.

[11] R. V. Guha, V. Gupta, V. Raghunathan, and R. Srikant. User modeling for a personal assistant. In *WSDM*, pages 275–284, 2015.

[12] J. Hullman, S. M. Drucker, N. H. Riche, B. Lee, D. Fisher, and E. Adar. A deeper understanding of sequence in narrative visualization. *TVCG*, 19(12):2406–2415, 2013.

[13] S. Idreos, O. Papaemmanouil, and S. Chaudhuri. Overview of data exploration techniques. In *SIGMOD*, pages 277–281, 2015.

[14] A. Kumar, R. McCann, J. F. Naughton, and J. M. Patel. Model selection management systems: The next frontier of advanced analytics. *SIGMOD Record*, 44(4):17–22, 2015.

[15] N. Labroche, M. Detyniecki, and T. Bärecke. Accelerating one-pass clustering by cluster selection racing. In *ICTAI*, pages 491–498, 2013.

[16] C. Lemke, M. Budka, and B. Gabrys. Metalearning: a survey of trends and technologies. *Artif. Intell. Rev.*, 44(1):117–130, 2015.

[17] V. Markl, V. Raman, D. E. Simmen, G. M. Lohman, and H. Pirahesh. Robust query processing through progressive optimization. In *SIGMOD*, pages 659–670.

[18] T. Milo and A. Somech. Next-step suggestions for modern interactive data analysis platforms. In *KDD*, pages 576–585, 2018.

[19] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *SIGKDD*, pages 1135–1144, 2016.

[20] S. Sarawagi. Explaining differences in multidimensional aggregates. In *Proceedings of VLDB*, pages 42–53, 1999.

[21] S. Sarawagi. User-adaptive exploration of multidimensional data. In *Proceedings of VLDB*, pages 307–316, 2000.

[22] V. S. Sengar and J. R. Haritsa. PLASTIC: reducing query optimization overheads through plan recycling. In *SIGMOD*, page 676, 2003.

[23] A. Simitsis, G. Koutrika, and Y. E. Ioannidis. Précis: from unstructured keywords as queries to structured databases as answers. *VLDB J.*, 17(1):117–149, 2008.

[24] K. Stefanidis, G. Koutrika, and E. Pitoura. A survey on representation, composition and application of preferences in database systems. *TODS*, 36(3):19:1–19:45, 2011.

[25] Q. Sun and B. Pfahringer. Pairwise meta-rules for better meta-learning-based algorithm ranking. *Machine Learning*, 93(1):141–161, 2013.

[26] P. Vassiliadis and P. Marcel. The road to highlights is paved with good intentions: Envisioning a paradigm shift in OLAP modeling. In *DOLAP*, 2018.

[27] P. Vassiliadis, P. Marcel, and S. Rizzi. Beyond roll-up's and drill-down's: An intentional analytics model to reinvent OLAP. *Submitted to Inf. Syst.*, 2019.

[28] A. Wasay, M. Athanassoulis, and S. Idreos. Queriosity: Automated data exploration. In *IEEE International Congress on Big Data*, pages 716–719, 2015.

[29] R. W. White. *Interactions with Search Systems*. Cambridge University Press, 2016.

[30] Z. Zhao, L. D. Stefani, E. Zgraggen, C. Binnig, E. Upfal, and T. Kraska. Controlling false discoveries during interactive data exploration. In *SIGMOD*, pages 527–540, 2017.

[31] M. Zuluaga, A. Krause, and M. Püschel. e-pal: An active learning approach to the multi-objective optimization problem. *JMLR*, 17:104:1–104:32, 2016.

[2] https://github.com/TAU-DB/REACT-IDA-Recommendation-benchmark