Bulletin of the Technical Committee on

Data Engineering

September 2018 Vol. 41 No. 3 IEEE Computer Society

Letters

Special Issue on Insights and Explanations in Data Analysis

News: TCDE 2018 Awardee Statements

News: TCDE Election

Conference and Journal Notices

The TC on Data Engineering

Membership in the TC on Data Engineering is open to all current members of the IEEE Computer Society who are interested in database systems. The TCDE web page is http://tab.computer.org/tcde/index.html.

The Data Engineering Bulletin

The Bulletin of the Technical Committee on Data Engineering is published quarterly and is distributed to all TC members. Its scope includes the design, implementation, modelling, theory and application of database systems and their technology.

Letters, conference information, and news should be sent to the Editor-in-Chief. Papers for each issue are solicited by and should be sent to the Associate Editor responsible for the issue.

Opinions expressed in contributions are those of the authors and do not necessarily reflect the positions of the TC on Data Engineering, the IEEE Computer Society, or the authors' organizations.

The Data Engineering Bulletin web site is at http://tab.computer.org/tcde/bull_about.html.

Letter from the Editor-in-Chief

News

I want to draw your attention to a new section of the Bulletin, the "News" section. There was enough happening to warrant a section of its own. I wanted to direct the reader's attention to this more than was possible by simply including additional letters.

TCDE AWARD WINNERS: Every year at the International Conference on Data Engineering, the Technical Committee on Data Engineering presents awards to individuals who have distinguished themselves in a particularly noteworthy way, through impact, service, education, or as an accomplished young researcher. This issue of the Bulletin provides the award winners with an opportunity to share their views on having received the award and describe the work that led to their receiving this distinction. Each awardee statement has a different "flavor", but all tell great stories. I enjoyed reading them and can recommend them to you.

TCDE CHAIR ELECTION: The chair of the Technical Commitee on Data Engineering serves a two year term. Thus, every other year, the TC holds an election for chair. It is the chair's responsibility to appoint the executive committee, whose members you see listed on the inside front cover. Xaiofang Zhou, the current chair, is precluded by a two term limit from seeking re-election to a third consecutive term. We have two distinguished and experienced candidates, Thomas Risse and Erich Neuhold, who have agreed to run for and serve as chair if elected. I would urge you all to express your preference by voting in this election.

The Current Issue

Data is everywhere. And it is increasingly valuable as we find improved ways of dealing with it. Indeed, data has been referred to as "the new oil". There is a worthwhile analogy here. Oil gushed from the ground spontaneously long before our modern era, and was largely wasted. Whether gigabytes through exabytes or more... it is not enough to simply have data. One needs to derive value from data.

As a technical community, we need to distill meaning from data. But perhaps more importantly, we need to provide technology that others (application experts) can use to distill meaning. Going a step further, we need to construct tools that can place the meaning in context, evaluate the strength of the analysis, and determine whether we should believe it. And the list goes on! This is challenging. And that is the focus of the current issue that Alexandra Meliou has put together for us. Alexandra did a great job in identifying people and their work that makes this Bulletin issue invaluable to not just members of our technical community but to anyone trying to understand how data is used and what it means. Thank you Alexandra.

David Lomet
Microsoft Corporation

Letter from the Special Issue Editor

The field of data analytics has flourished and continues to grow with unabated momentum. Insights derived from rich and diverse data sets drive business decisions and modern software. The race to derive more insights, faster, and more effectively has intensified, and research has followed suit. At the same time, data and analytics tools have grown complex, and analysis results are often poorly understood. In this issue, we bring together an exciting collection of recent and ongoing work, investigating methods for accelerating the derivation of insights in data analysis and technologies for understanding the insights derived from data.

We start with "The Case for a Visual Discovery Assistant: A Holistic Solution for Accelerating Visual Data Exploration", by Doris Jung-Lin Lee and Aditya Parameswaran. The paper considers the use of visualizations in exploratory data analysis, and presents a vision for automating the navigation across large collections of visualizations. It provides a review of existing and ongoing work, and outlines open research questions and challenges as we are moving towards the direction of expecting less input from the human users and the system bears more of the onus in visual discovery.

In the second paper, "Towards Quantifying Uncertainty in Data Analysis & Exploration", Yeounoh Chung, Sacha Servan-Schreiber, Emanuel Zgraggen, and Tim Kraska consider the many facets of uncertainty and its impact on insights derived from analytics and exploration pipelines. The paper presents a toolset for quantifying uncertainty with the objective of making data analysis reliable and safe, guarding against incorrect analyses and false discoveries.

The third paper focuses on methods for understanding the validity of data insights. In "Query Perturbation Analysis: An Adventure of Database Researchers in Fact-Checking", Jun Yang, Pankaj K. Agarwal, Sudeepa Roy, Brett Walenz, You Wu, Cong Yu, and Chengkai Li point out that even though data can be used to back a claim, that claim may still be misleading and essentially incorrect. Discovering the query that produced the claim and perturbing it appropriately can shed light on the claim's validity. The paper highlights results in query perturbation and computational fact checking that largely predate the rise of the so-called "fake news", as well as practical deployments of the relevant systems, including during the 2016 US federal elections.

The next paper, "Explanation Tables", by Kareem El Gebaly, Guoyao Feng, Lukasz Golab, Flip Korn, and Divesh Srivastava, considers deriving data summaries; such summaries can aid human understanding of a dataset's properties. The paper focuses on the problem setting of a relation with categorical dimension attributes and one binary outcome attribute, where the objective is to derive an interpretable explanation of the factors affecting the binary attribute.

In "Ontology-Based Natural Language Query Interfaces for Data Exploration", Chuan Lei, Fatma Özcan, Abdul Quamar, Ashish Mittal, Jaydeep Sen, Diptikalyan Saha, and Karthik Sankaranarayanan argue for the use of natural language as the query interface to data exploration. Exploratory tasks are varied and often require different querying capabilities of the underlying data stores. Natural language can become a unifying interface, bypassing the need for users to learn and use other complex languages, thus increasing the accessibility of analytics. The paper describes an end-to-end NLQ system, overviews ongoing work and challenges, and highlights practical use cases.

We conclude with a paper that reexamines a fundamental component of data processing pipelines: data structures. "The Periodic Table of Data Structures", by Idreos et al., points out that data structure decisions impact the types of analysis that one can perform, the efficiency of that analysis, and, ultimately, the insights that can be achieved. The paper describes a vision that calls for breaking data structures into first principles, reexamining the design space, speculatively estimating a data structure's impact on a workload, and automating data structure design to cater to specific analysis tasks.

Thank you to all the authors for their insightful contributions, and thank you to Dave Lomet for valuable guidance on putting together the issue. I hope you enjoy this collection.

Alexandra Meliou
University of Massachusetts Amherst

The Case for a *Visual Discovery Assistant*: A Holistic Solution for Accelerating Visual Data Exploration

Doris Jung-Lin Lee, Aditya Parameswaran
{jlee782,adityagp}@illinois.edu
University of Illinois, Urbana-Champaign

Abstract

Visualization is one of the most effective and widely-used techniques for understanding data. Yet, the growing use of visualizations for exploratory data analysis poses new demands beyond simply the graphical presentation and visualization authoring capabilities offered in existing tools. In particular, many data analysis tasks involve navigating large collections of visualizations to make sense of trends in data; at present, this navigation is done manually or programmatically. We outline a vision for an intelligent, interactive, understandable, and usable tool that can help automate this largely manual navigation: we call our tool VIDA[1]*—short for VIsual Discovery Assistant. We argue that typical navigation tasks can be organized across two dimensions—overall goal and precision of specification. We organize prior work—both our own work, as well as other ongoing work in this area—across these two dimensions, and highlight new research challenges. Together, addressing these challenges underlying* VIDA *can help pave the way for a comprehensive solution for removing the pain points in visual data exploration.*

1 Introduction

With the ever-increasing complexity and size of datasets, there is a growing demand for information visualization tools that can help data scientists make sense of large volumes of data. Visualizations help discover trends and patterns, spot outliers and anomalies, and generate or verify hypotheses. Moreover, visualizations are accessible and intuitive: they tell us stories about our data; they educate, delight, inform, enthrall, amaze, and clarify. This has led to the overwhelming popularity of point-and-click visualization tools like Tableau [33], as well as programmatic toolkits like ggplot, D3, Vega, and matplotlib. We term these tools as *visualization-at-a-time* approaches, since data scientists need to individually generate each visualization (via code or interactions), and examine them, one at a time.

As datasets grow in size and complexity, these visualization-at-a-time approaches start to break down, due to the limited time availability on the part of the data scientists—there are often too many visualizations to examine for a given task, such as identifying outliers, or inferring patterns. Even on a single table, visualizations can be generated by varying the subsets of data operated on, or the attributes (or combinations thereof) that can be visualized. If we add in various visualization modalities, encodings, aesthetics, binning methods, and

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1] Vida means *life* in Spanish.

transformations, this space becomes even larger. (For this work, we focus primarily on the impact of varying the data subsets and attributes visualized—some of these latter concerns are the focus of recent work [39].)

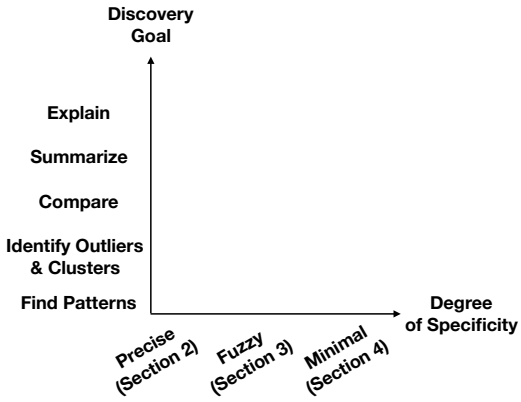


Figure 1: The two dimensions of discovery settings. The Y axis depicts the different discovery goals ordered in roughly increasing complexity, while the X axis characterizes decreasing levels of query specificity and correspondingly increasing levels of autonomous assistance.

Thus, there is a pressing need for an intelligent, interactive, understandable, usable, and enjoyable tool that can help data scientists navigate collections of visualizations, across a range of possible analysis goals and modalities that data scientists may require. We term our hypothesized comprehensive tool VIDA, short for *VIsual Discovery Assistant*. Data scientists can specify any discovery goal at a high level, in one of many intuitive input modalities supported in VIDA, with VIDA automatically traversing visualizations to provide solution or guidance for the specified discovery goal, thereby eliminating the tedium and wasted labor of comparable visualization-at-a-time approaches.

VIDA Dimensions. In order to be a holistic solution for navigating collections of visualizations, VIDA must be able to support various discovery settings. We organize these settings along two dimensions, displayed along the Y axis and X axis (respectively) of Figure 1—first, the overall discovery goal, and second, the degree of specificity of the input query.

Along the Y axis, we identify five common discovery goals in visual data exploration: *finding patterns*, *identifying anomalies/clusters*, *summarizing*, *performing comparisons*, *providing explanations*. These five goals borrow from functionalities in existing systems, as well as related visualization task taxonomies [2, 11]. They are organized along the Y axis in a sequence of roughly increasing complexity; however, we must emphasize that these goals are distinct from one other. We omit low-level goals such as filtering or sorting, since these functionalities are common in existing visualization-at-a-time tools and toolkits. We also omit goals that go beyond visual data exploration, such as extrapolation, supervised learning, and cleaning, among others.

We identify three degrees of specificity for the query input, organized along the X axis: *precise*, *fuzzy*, *minimal*. The degree of specificity characterizes the division of work between how much user has to specify versus how much the system has to automatically infer and aid in accomplishing the discovery goal. For the precise setting, the onus is placed on the user to provide an exact and complete specification of what the solution to their discovery goal must look like; for the fuzzy setting, the user can provide a vague specification of what the solution must look like; and finally, for the minimal setting, the user provides a minimal specification, or leaves the characteristics of the solution underspecified, leaving it up to the system to "fill in" the rest. As we proceed along the X axis, it gets harder for the system to automatically interpret what the user might have in mind as a solution to their goal.

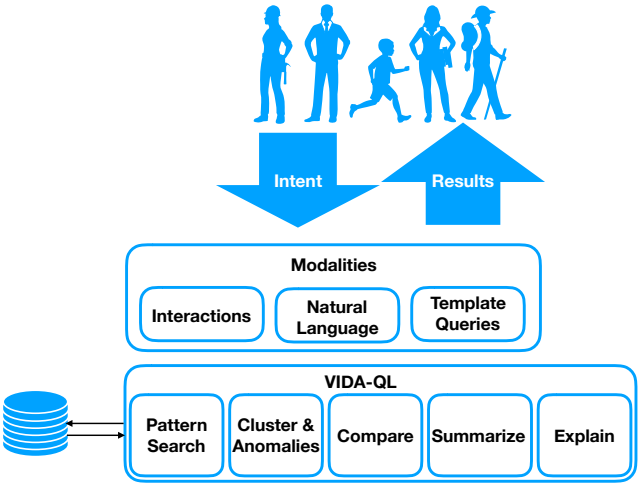


Figure 2: VIDA Architecture

VIDA Input Modalities. To support the spectrum of demands imposed by the discovery settings described above, VIDA must support a range of interactive input modalities, as displayed in Figure 2, catering to a range of user expertise and preferences. These input modalities include:

- restricted template queries, involving selection of operators and operands from a drop-down menu; and

VQS	Discovery Goals Covered	Specificity	Interactions Supported
Zenvisage [18, 31]	Find Patterns, Cluster/Outliers, Compare	Precise	Interactions (Clicking, Sketch, Drag-and-drop), Query Language
TimeSearcher [12]	Find Patterns, Cluster/Outliers	Precise	Interactions (Clicking, Sketch)
Query-by-Sketch [36]	Find Patterns	Precise	Interactions (Clicking, Sketch)
Scorpion [41]	Explain	Fuzzy	Interactions (Clicking)
ShapeSearch [32]	Find Patterns	Fuzzy	Interactions (Sketch), Natural Language, Query Language
Profiler [15]	Compare	Fuzzy	Interactions (Clicking, Brushing-and-Linking)
SeeDB [34]	Compare	Minimal	Interactions (Clicking)
Storyboard [17]	Cluster/Outliers, Compare, Summarize	Minimal	Interactions (Clicking)
iDiff [27, 28]	Compare, Explain	Minimal	Query Language

Table 1: A Sample of Visual Query Systems

- interactions, either with an existing visualization, such as brushing-and-linking or hovering, or those that construct a new one, such as drag-and-drop or sketching; and
- natural language, via a keyword search box, dialog, or conversational interface.

Each of these inputs are compiled down into a query in a query language, called VIDA-QL. Alternatively, expert users may directly invoke VIDA-QL queries. VIDA-QL queries will natively support the five discovery goals and combinations thereof, e.g., summarization followed by pattern search. Another important element is how much does a user actively requests for visualizations ("pull") versus how much VIDA recommends visualizations ("push"). Given that we expect VIDA to support a range of specificities, VIDA must support both push and pull, with pull decreasing in importance and push gaining importance, as we go from the precise to minimal setting.

Present Day Visual Query Systems. We have described VIDA so far as if no comparable capabilities exist today. This is not the case; in Table 1, we list some examples of systems that partially provide the functionality of VIDA, for certain settings and input modalities. We will describe some of these systems later on in the paper. Since all of these systems allow users to query data *visually*, we call these systems *Visual Querying Systems*, or VQSs for short. VQSs typically *(i)* employ objectives that are perceptually-aware, taking into account for the fact that the results are typically consumed by a human analyst, rather than an algorithm; *(ii)* provide some interactive interface or declarative capabilities to express the discovery goal as a "what" rather than a "how"; and *(iii)* possess optimization capabilities to facilitate the efficient discovery of visual insights.

Outline. The rest of our paper is organized along the degree of specificity of discovery goal, and we will allude to the specific discovery goals as well as input modalities as we go along. The degree of input specificity is the factor that most significantly affects the architecture of VIDA, with the complexity increasing as the specificity decreases—in that the system has to do more "guesswork" to support underspecified goals.

We begin by discussing the the *precise* setting (Section 2). We describe ZENVISAGE [31] as a partial solution for this setting, and thereby a starting point for VIDA, partially eliminating the problem of having to manually examine large numbers of visualizations for a given discovery goal, which can be error-prone and inefficient.

However, a design study using ZENVISAGE demonstrates that the precise setting is insufficient for addressing all of the demands of real-world use-cases [18]. In particular, users do not have a clear understanding of their querying intent without looking at example visualizations or summaries of the data, and even when they do, their intent involves vague or high-level notions that can not be expressed clearly.

To bridge the gap between the user's high-level intent and the system demands, we outline a set of research challenges that goes beyond simple precise visual querying, by (a) supporting a wider class of vague or "fuzzy" high-level queries, thereby increasing the expressiveness of VIDA (Section 3); and (b) making it easier to know what to query by recommending visualizations that provide a high-level understanding of the data (Section 4).

Interfaces and Interactions, not Optimization. In this paper, we focus our attention on the interfaces, query languages, and capabilities, and how to use them to capture discovery goals with varying degrees of specification, as opposed to optimization. Indeed, once the query is issued in some form (say, translated into a VIDA-QL query), optimization is crucial to try to traverse the space of visualizations and return results in an interactive manner. A range of techniques may be adopted for this purpose, including parallelism, multi-query optimization,

sampling, pruning, and pre-computation [31, 34]. We expect a similar combination of techniques to be applicable to each of the settings that we consider. To limit the scope of this paper, we have decided to avoid describing optimization aspects and focus on the interfaces and interactions instead.

2 The Precise Setting: Existing Visual Query Systems

While visual data exploration often reveals important anomalies or trends in the data [11, 22], it is challenging to use visualization-at-a-time systems to repeatedly choose the right subsets of data and attributes to visualize in order to identify desired insights. We first motivate the precise setting through a use-case from astronomy.

2.1 Motivating Example: Discovering Patterns in Astronomical Data

Astronomers from the Dark Energy Survey (DES) are interested in finding anomalous time series in order to discover transients, i.e., objects whose brightness changes dramatically as a function of time, such as supernova explosions or quasars [7]. When trying to find celestial objects corresponding to supernovae, with a specific pattern of brightness over time, astronomers individually inspect the corresponding visualizations until they find ones that match the pattern. With more than 400 million objects in their catalog, each having their own set of time series brightness measurements, manually exploring so many visualizations is not only error-prone, but also overwhelming. Many astronomers instead rely on guess-work or prior evidence to explore the visualizations, rather than directly "searching" for the patterns that they care about. While most astronomers do use programming tools, such as Python and R, it is cumbersome to express each new information need via carefully hand-optimized code. The astronomy use case highlights a common challenge in visual data exploration, where there are often a large number of visualizations for each discovery goal, and manual exploration is impossible. There is no systematic way to create, compare, filter, and operate on large collections of visualizations.

2.2 Ongoing Work and Research Challenges within the Precise Setting

Discovering Patterns and Anomalies/Clusters, and Performing Comparisons with ZENVISAGE. ZENVISAGE is an example of a VQS that operates on large collections of visualizations. ZENVISAGE is built on top of a visual query language called ZQL, which operates on collections of visualizations, and returns collections of visualizations, using primitives such as visualization composition, sorting, and filtering [31]. In addition, via user-defined functions (also provided as built-ins), ZQL can compute the distance for a visualization from another—and this distance can be used as a building block for more complex operations, such as the first three discovery goals listed in Table 1, i.e., finding a visualization matching a pattern, detecting an outlier or cluster centers, or comparing visualizations with each other. Thus, ZQL operates at a level higher than languages for specifying visual encodings of individual visualizations [33, 37]. ZQL can be used to construct a rich variety of queries, including the following on a real-estate dataset:

- *Find Patterns.* Find visualizations of cities whose housing price over time is similar to Manhattan.
- *Identify Anomalies.* Find visualizations of cities with both an increasing housing price trend and an increasing foreclosure rate trend.
- *Perform Comparisons.* Find visualizations for which New York and Champaign differ the most.
- *Find Patterns, followed by Clusters.* For cities that are similar to New York on sales price trends, find typical trends for foreclosure rates.

While ZQL is a useful starting point, writing ZQL queries can be daunting for users who are not comfortable with programming. Therefore, we extracted a typical workflow of visual querying for finding patterns, identifying outliers and clusters, and making comparisons, and made it expressible via simple interactions. As shown

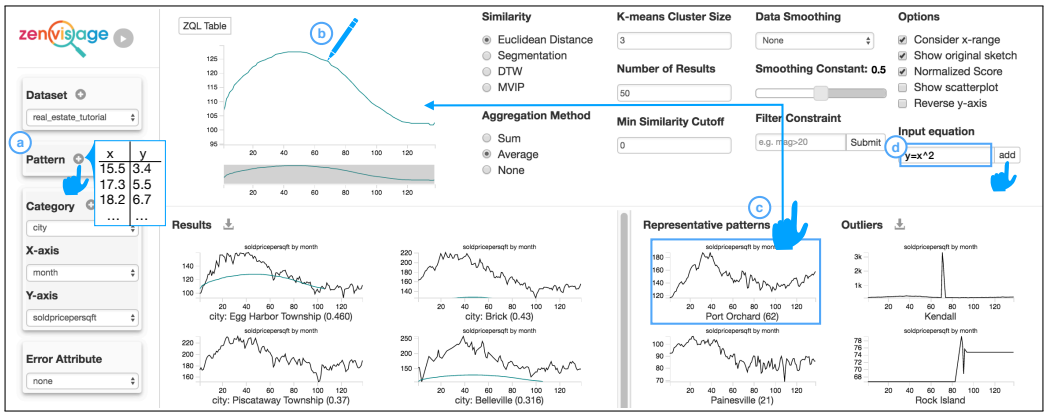


Figure 3: ZENVISAGE offers a variety of querying modalities, including: a) uploading a sample pattern from an external dataset as a query, b) sketching a query pattern, c) dragging-and-dropping an existing pattern from the dataset, and d) inputting an equation as a query, from [18].

in Figure 3, the user can input their search pattern via (a) inputting a set of points, (b) a sketch, (c) dragging-and-dropping an existing visualization, or (d) inputting an equation, and specify a space of visualizations over which to search. The system returns a ranked list of matching visualizations, shown in the "Results" pane below the canvas in (b), for the corresponding ZQL query. The system also provides typical trends (called "Representative Patterns", highlighted under (c)) and outliers for additional context. The user can also switch to a tabular interface wherein they can input ZQL queries. Users can also make comparisons between different subsets of the data by examining visualizations of dynamically created classes in ZENVISAGE [18]. Thus, this workflow captures one variation of the three discovery goals—finding patterns, identifying outliers/clusters, and making comparisons.

Other Related Precise Visual Querying Systems. There has been a range of work primarily focused on locating visualizations matching certain patterns, including the influential work on TimeSearcher [12], where the query is specified as a box constraint, and Query-by-Sketch [36], where the query is sketched as a pattern. This was applied to search queries within Google Correlate [21]. Recent work has also extended this to querying using regular expressions [42], while others have analyzed the semantics of sketches [5, 19].

Limitations within the Precise Setting. The most important challenge within the precise setting is the following: *How do we expose the entire capabilities of* VIDA-QL *via intuitive interactions and discoverable modules?* So far, most tools that we listed above support a restricted set of interactions, primarily focused on finding patterns via sketches or box constraints. ZENVISAGE extends and builds on this by supporting outlier and representative trend discovery from the front-end interface. Unfortunately, performing comparisons, deriving explanations, or summaries, is hard to do interactively. In fact, deriving explanations or summaries is hard to do even with the full power of ZQL, since it does not support the synthesis of new information that goes beyond simple processing of collections of visualizations. It remains to be seen if and how ZQL can be extended to meet these capabilities. Moreover, even for sketching, it would be interesting to see if we can support analogous pattern search mechanisms for other visualization types, such as scatterplots or heatmaps.

2.3 Limitations with the Precise Setting

Even if we could address the issues within the precise setting in the previous section, there are some fundamental limitations with the precise setting itself that need to be addressed within a full-fledged VIDA. We discuss these limitations in the context of a participatory design process for the development of ZENVISAGE, in collaboration with scientists from astronomy, genetics, and material science [18].

The Problem of Interpreting Ambiguous, High-level Queries. When users interact with ZENVISAGE, they often translate their ambiguous, high-level questions into an plan that consists of multiple interactions to incre-

mentally address their desired query. The expressiveness of such a system comes from the multiplicative effect of stringing together combinations of interaction sequences into a customized workflow. Designing features that diversify potential variations expands the space of possible workflows that could be constructed during the analysis. However, even with many supported interactions, there were still vague and complex queries that could not be decomposed into a multi-step interaction workflow. For example, ZENVISAGE was unable to support high-level queries that involved the use of vague descriptors for matching to specific data characteristics, such as finding patterns that are 'flat and without noise', or 'exhibits irregularities'. These scenarios showcase examples of lexical ambiguity, where the system can not map the vague term 'irregular' or 'noisy' into the appropriate series of analysis steps required to find these patterns. In Section 3, we survey the challenges in supporting vague and complex queries and point to some ongoing research.

The Problem of Not Knowing What to Query. Another key finding is that users often do not start their analysis with a specific pattern in mind or a specific comparison to perform. For example, within ZENVISAGE, many users first made use of the recommended representative trends and outlier visualizations as contextual information to better understand their data, or to query based on these recommended visualizations. Thus, the recommended visualizations are often used to form queries in a bottom-up manner, rather than users coming up with a query in a top-down, prescriptive manner. Thus, there is a need for visualization recommendations [35] that can help users jump-start their exploration. Recommendations help facilitate a smoother flow of analysis by closing the gap between the two modalities of querying and exploration, reminiscent of the browsing and searching behaviors on the Web [24], thus ensuring that user is never stuck or out of ideas at any point during the analysis. Typically, visualization recommendations help accelerate the process of discovering interesting aspects of the data by broadening exploration. In Section 4, we argue that recommendations should not just focus on increasing breadth-wise exploration so that other views of the data are discoverable, but they should also contribute towards helping users become more aware of the distributions and trends in the data, as well as the broader context of their analysis.

3 The Fuzzy Setting: Towards Intelligent Visual Search

As we argued in the previous section, there are ambiguous, high-level goals (e.g., "explain this bump in this visualization", or "find a product for which the visualization of sales over time is bumpy") that cannot be captured within the precise setting. We will discuss this fuzzy setting in the following section.

3.1 The Challenge of Usability-Expressiveness Tradeoff

The challenge for supporting ambiguous, high-level goals stems from the inevitable design trade-off between query expressiveness and interface usability in interactive data exploration systems [14, 22]. This tradeoff is observed not only in visual data exploration systems, but also true for general ad-hoc data querying. While querying language such as SQL are highly expressive, formulating SQL queries that maps user's high-level intentions to specific query statements is challenging [14, 16]. As a result, query construction interfaces have been developed to address this issue by enabling direct manipulation of queries through graphical representations [1], gestural interaction [23], and tabular inputs [8, 43]. For example, form-based query builders often consist of highly-usable interfaces that ask users for a specific set of information mapped onto a pre-defined query. However, form-based query builders are often based on query templates with limited expressiveness in their semantic and conceptual coverage, which makes it difficult for expert users to express complex queries. The extensibility of these systems also comes with high engineering costs, as well as potentially overwhelming users with too many potential options to chose from. Thus, there is a need for tools that enable users to formulate rich and complex queries, yet amenable to the fuzzy, imprecise query inputs that users might naturally formulate.

3.2 Ongoing Work and Research Challenges within the Fuzzy Setting

Given the tradeoff between expressiveness and usability, we discuss a growing class of VQSs that targets how to answer imprecise, fuzzy, and complex queries. In the fuzzy setting, the most important challenge is *how do we interpret fuzzy, complex queries, and allow users to understand the results and refine the analysis*. This boils down to several challenges that we will discuss in this section, including: *How can we develop better ways to resolve ambiguity by inferring the information needs and intent of users? What is the appropriate level of feedback and interactions for query refinement? How can we develop interpretable visual metaphors that explain how the query was interpreted and why specific query results are returned?*

We organize our discussion along the types of ambiguity that may arise in interpreting fuzzy, complex queries. Since systems targeting the fuzzy setting operate in the intermediate layer between users and VIDA-QL as shown in Figure 2, we use the linguistic classification scheme to provide an analogy for where the ambiguous aspects of queries may arise, noting that the use of this analogy by no means limits our analysis to only natural language interfaces.

Lexical Ambiguity: Lexical ambiguity involves the use of vague descriptors in the input queries. Resolving these lexical ambiguities has been a subject of research in natural language interfaces for visualization specification[2], such as DataTone [10] and Eviza [30]. These interfaces detect ambiguous quantifiers in the input query (e.g. "Large earthquakes near California"), and then displays ambiguity widgets in the form of a widget to allow users to specify the definition of 'large' in terms of magnitude and the number of miles radius for defining 'near'. These ambiguity widgets not only serve as a way to provide feedback to the system for lexically vague queries, but is also a way for explaining how the system has interpreted the input queries. It remains to be seen if similar ambiguity-resolution techniques can be applied for exploration of collections of visualizations as opposed to one visualization at a time.

In addition to ambiguity in the filter descriptions, as we have seen in the ZENVISAGE study, there is often also ambiguity in the terminologies used for describing query patterns. SHAPESEARCH [32] operates on top of an internal shape query algebra to flexibly match visualization segments, such as a trendline that "first rises and then go down". After the translation to the internal query representation, SHAPESEARCH then performs efficient perceptually-aware matching of visualizations. SHAPESEARCH is restricted to trendline pattern search, without supporting the other four discovery goals. Similarly, within VIDA, we envision lexical ambiguity to be resolved by determining the appropriate *parameters* to the internal VIDA-QL query for achieving the user's desired querying effects.

Syntactic Ambiguity: Syntactic ambiguity is related to the vagueness in specifying how the query should be structured or ordered. For example, DataPlay introduced the idea of syntax non-locality in SQL, in which switching from an existential (at least one) to a universal (for all) quantifier requires major structural changes to the underlying SQL query [1]. DataPlay consists of a visual interface that allowed users to directly manipulate the structure of the query tree into its desired specification. Within VIDA, syntactic ambiguities resolution h involves mapping portions of the vague queries into to *a series of multi-step workflows* to be executed in VIDA-QL and exposing feedback frameworks to allow users to directly tweak the internal query representation.

Semantic Ambiguity: Semantic ambiguity includes addressing 'why' questions that are logically difficult to answer, such as 'why do these datapoints look different from others?'. For example, Scorpion [41] visually traces the provenance of a set of input data tuples to find predicates that explain the outlier behavior; it remains to be seen if this work, along with others on causality, e.g., [20, 26], can be exposed via front-end interactions or natural language queries. Semantic ambiguity can also arises when the user does not specify their intent completely or explicitly, which is often the case in early stages of visual data exploration. Evizeon [13], another natural language interface for visualization specification, makes use of anaphoric references to fill in incomplete follow-on queries. For example, when a user says 'Show me average price by neighborhood', then 'by home

[2]These are not VQSs, since they do not operate on collections of visualizations, rather, they expect one visualization to be the result for a query. Thus, they act as a natural language interface for visualization-at-a-time systems.

type', the system interprets the anaphoric reference as continuing the context of the original utterance related to average price on the y-axis. Semantic ambiguity can often be composed of one or more lexical and syntactical ambiguities. For example, in Iris [9], a dialog system for data science, users can specify a vague, high-level query such as 'Create a classifier', then Iris makes use of nested conversations to inquire about what type of classifiers and features to use for the model, to fill in the details of the structure and parameters required.

4 The Minimal Setting: Towards Recommendations for Data Understanding

While our focus in the previous sections have been on intent-driven queries, where users have some knowledge of the types of visual queries they may be interested in, one of the key goals of visual data exploration is to promote a better understanding of the dataset to enable users to make actionable decisions. Within the minimal setting, we target systems that help users become more aware of their dataset and visualize where they are in their analysis workflow with minimal required user input. This can often be useful when there is an absence of explicit signals from the user, such as when a user is at the beginning of their analysis (commonly known as the 'cold-start' problem) or when the user does not know what to query for. We will first describe SEEDB and STORYBOARD as examples of visual query systems that suggest informative data views for jumpstarting further analysis. Then, we will discuss the importance of contextual awareness during dynamic visual data exploration to highlight the challenges and opportunities ahead in this space.

SEEDB: Exploring Multi-variate Data Dimensions Through Interesting Visualization Recommendations. Identifying interesting trends in high-dimensional datasets is often challenging due to the large combination of possible X,Y axes for the generated visualizations. To perform this task, users need to manually create, examine, and compare relationships between these multi-variate visualizations to discover interesting insights.

Within SEEDB [34], a user can specify a subset of data that they would be interested in exploring. Then, SEEDB searches for visualizations (i.e., specific combinations of X and Y axes) that show how this subset differs from the rest of the data, and are therefore *interesting*, and provides these recommendations within a side panel in the visual exploration interface. The evaluation user study showed that SEEDB-recommended visualizations are three times more likely to be interesting compared to manually constructed visualizations and provide an effective starting point for further analysis. Profiler [15] provides similar recommendations targeting the identification of anomalies in data.

STORYBOARD: Promoting Distribution Awareness of Data Subsets with Summary of Visualizations. Whereas SEEDB looks at the space of possible X and Y axes for a given data subset, our recent system STORYBOARD explores the space of possible subsets (or subpopulations) of data, for fixed X and Y axes. Common analytics tasks, such as causal inference, feature selection, and outlier detection, require understanding the distributions and patterns present in the visualizations at differing levels of data granularity [3, 11, 41]. However, it is often hard to know *what* subset of data contains an insightful distribution to examine. In order to explore different data subsets, a user would first have to construct visualizations corresponding to all possible data subsets, and then navigate through this large space of visualizations to draw meaningful insights.

STORYBOARD is an interactive visualization summarization system that automatically selects a set of visualizations to characterize and summarize the diversity of distributions within the dataset [17]. Figure 4 illustrates an example dashboard generated by STORYBOARD from the Police Stop Dataset [25], which contains records of police stops that resulted in a warning, ticket, or an arrest. STORYBOARD was asked to generate a dashboard of 9 visualizations with x-axis as the stop outcome and y-axis as the percentage of police stops that led to this outcome. First, at the top of our dashboard, STORYBOARD highlights three key data subsets that result in a high arrest rate, which looks very different than the overall distribution at the root node (where the majority of stops results in tickets). Following along the leftmost branch, we learn that even though in general when a search is conducted, the arrest rate is almost as high as ticketing rate, when we look at the Asian population, whether a search is conducted had less influence on the arrest rate and the trend more resembles the overall distribution.

STORYBOARD makes use of a context-dependent objective to search for visualizations within the data subset lattice for which *even the informative parents—i.e., the parent visualization within the data subset lattice that is closest to the present visualization—fail to accurately predict or explain the visualization.*

The effectiveness of STORYBOARD largely comes from how the summary visualizations help users become more distributionally aware of the dataset. We define *distribution awareness* as the aspect of data understanding in which users make sense of the key distributions across different data subsets and their relationships in the context of the dataset. With distribution awareness, even though it may be infeasible for an user to examine all possible data subsets, the user will still be able to draw meaningful insights and establish correlations about related visualizations by generalizing their understanding based on the limited number of visualizations presented in the dashboard. Our evaluation study shows that facilitating distribution awareness through STORYBOARD guides users to make better predictions regarding unseen visualizations, ranking attribute importance, and retrieval of interesting visualizations compared to dashboards generated from the baselines.

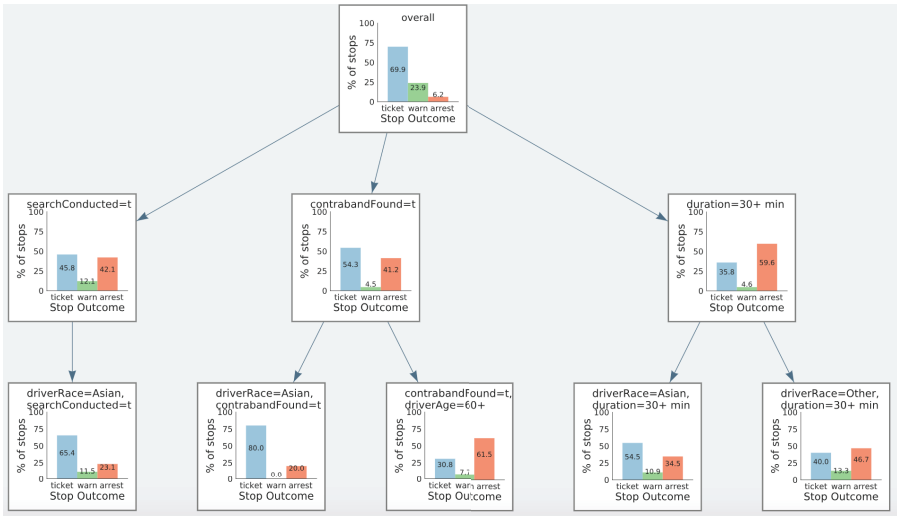


Figure 4: Example dashboard generated by STORYBOARD summarizing the key insights in the Police dataset.

4.1 Research Challenges within the Minimal Setting: Distributional to Contextual Awareness

The main research challenge that we aim to address in the minimal setting is the following: *How can contextual information be used to help facilitate better understanding, and guide users towards more informative next steps in their analysis?* The notion of distribution awareness is useful when considering the scenario at one static point in time of the analysis, such as during cold-start. In this section, we introduce a complementary notion of data understanding called *contextual awareness*, which is essential when considering a dynamic analytic workflow for visual data exploration.

Contextual awareness is the aspect of data understanding related to the *situation* (what is the information that I'm currently looking at and how did it come about?) and *history* (what have I explored in the past and where should I look next?) of exploration. Situational understanding involves recognizing what data is in the current scope of analysis, including making sense of the data attributes and schema and keeping track of what filters or transformations have been applied to the displayed data. Historical understanding is associated with the user's past analysis actions on the data. As an example, a user may be interested in how the sales price of a product changes as a function of other dimensions variables, such as geographic location, year sold, and product type. Situational information informs them that they are looking at a bar chart with X=TYPE, Y=AVG(PRICE), whereas historical information points to the fact that they should explore the geographic dimension, since they have already explored the temporal attribute YEAR.

While the problem of data provenance has been well studied in database literature [4, 6, 40], the effects of showing provenance information to users during data analysis is an important but underexplored area. Moreover, within a dataset, history is essential in helping users navigate through the space of possible analysis actions and provide users with sense of coverage and completion. The notion of adding navigational cues to guide

exploration in visual information spaces was first proposed in Willet et al.'s work on *scented widgets* [38]. In Pirolli and Card's theory of information foraging, scents are cues that signifies the perceived benefit that one would receive during a search. Scented widgets adds to existing search interfaces by embedding visualizations that provide informational scents, such as histogram distributions of how popular a particular value is among users or using color to encode the size of a dataset in a drop-down menu. Recently, Sarvghad et al. have extended the idea of scented widgets to incorporate dimension coverage information during data exploration, including which dimensions have been explored so far, in what frequency, and in which combinations [29]. Their study shows that visualizing dimension coverage leads to increased number of questions formulated, findings, and broader exploration. Interpretable and non-disruptive cues that enables users to visualize provenance and history help sustain contextual awareness and guide users towards more informative next steps in their analysis.

Mechanisms that facilitate distribution awareness for users can effectively couple with contextual awareness in dynamic exploration situations to help update the user's mental model on the current data context. For example, the representative and outlier patterns in ZENVISAGE provides summaries of data in context. When a dataset is filtered, the representative trends are updated accordingly. By being aware of both the context and the distributions, the users becomes distributionally aware of how the typical patterns and trends of the distributions changes in a particular context.

We envision that by incorporating contextual information along with improving the recommendation ('pull') aspects of the VIDA-QL discovery modules, VIDA will be well-equipped with the necessary information for making 'active' recommendations. Contrary to 'static' recommendations in STORYBOARD and SEEDB, where users need to submit some minimal required information to explicitly request for recommendations and the system performs only one type of recommendation, these next-generation systems actively seek for opportunities to provide appropriate recommendations that would be useful to the user at the specific stage of analysis by intelligently adapting the modes of recommendation to the context of user's analysis. This shift from static to active recommendation is analogical to the shift from precise to fuzzy querying in Section 3, where the onus is more on the system to 'guess' at the users intent. In active recommendations, instead of providing minimal information to request recommendation, the system will automatically infer implicit signals through other modalities of interaction. VIDA can make use of this information to make better recommendations that can guide users towards meaningful stories and insights for further investigation.

5 Concluding Remarks

Data is inherently agnostic to the diverse information needs that any particular user may have. Visual data exploration systems can help bridge the gap between what users want to get from the data through querying and what insights the data has to offer through recommendations. To facilitate a more productive collaboration, in this paper, we outline our vision for VIDA, motivated by related work from precise visual querying of informative visualizations to accelerate the process of data discovery; to interpreting ambiguous and high-level queries through query refinement feedback; to recommendations that promote better distributional and contextual awareness for users. We hope that the agenda sketched out in this paper sheds light on the many more exciting research questions and opportunities to come in this nascent field.

References

[1] Azza Abouzied, J Hellerstein, and A Silberschatz. Dataplay: interactive tweaking and example-driven correction of graphical database queries. *Proceedings of the 25th annual ACM symposium on User interface software and technology*, pages 207–217, 2012.

[2] Robert Amar, James Eagan, and John Stasko. Low-level components of analytic activity in information visualization. *Proceedings - IEEE Symposium on Information Visualization, INFO VIS*, pages 111–117, 2005.

[3] Anushka Anand and Justin Talbot. Automatic Selection of Partitioning Variables for Small Multiple Displays. *IEEE Transactions on Visualization and Computer Graphics*, 22(1):669 – 677, 2015.

[4] Peter Buneman, Adriane Chapman, and James Cheney. Provenance management in curated databases. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data*, SIGMOD '06, pages 539–550, New York, NY, USA, 2006. ACM.

[5] Michael Correll and Michael Gleicher. The semantics of sketch: Flexibility in visual query systems for time series data. In *Visual Analytics Science and Technology (VAST), 2016 IEEE Conference on*, pages 131–140. IEEE, 2016.

[6] Y. Cui and J. Widom. Lineage tracing for general data warehouse transformations. *Proceedings of the VLDB Endowment*, May 2003.

[7] Drlica Wagner et al. Dark energy survey year 1 results: The photometric data set for cosmology. *The Astrophysical Journal Supplement Series*, 235(2):33, 2018.

[8] David W. Embley. Nfql: The natural forms query language. *ACM Transactions on Database Systems (TODS)*, June 1989.

[9] Ethan Fast, Binbin Chen, Julia Mendelsohn, Jonathan Bassen, and Michael S. Bernstein. Iris: A conversational agent for complex tasks. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 473:1–473:12, New York, NY, USA, 2018. ACM.

[10] Tong Gao, Mira Dontcheva, Eytan Adar, Zhicheng Liu, and Karrie G. Karahalios. Datatone: Managing ambiguity in natural language interfaces for data visualization. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*, UIST '15, pages 489–500, New York, NY, USA, 2015. ACM.

[11] Jeffrey Heer and Ben Shneiderman. Interactive Dynamics for Visual Analysis. *ACM Queue*, 10(2):30, 2012.

[12] Harry Hochheiser and Ben Shneiderman. Dynamic query tools for time series data sets: timebox widgets for interactive exploration. *Information Visualization*, 3(1):1–18, 2004.

[13] Enamul Hoque, Vidya Setlur, Melanie Tory, and Isaac Dykeman. Applying Pragmatics Principles for Interaction with Visual Analytics. *IEEE Transactions on Visualization and Computer Graphics*, (c), 2017.

[14] H. V. Jagadish, Adriane Chapman, Aaron Elkiss, Magesh Jayapandian, Yunyao Li, Arnab Nandi, and Cong Yu. Making database systems usable. SIGMOD '07, pages 13–24, New York, NY, USA, 2007. ACM.

[15] Sean Kandel, Ravi Parikh, Andreas Paepcke, Joseph Hellerstein, and Jeffrey Heer. Profiler: Integrated statistical analysis and visualization for data quality assessment. In *Advanced Visual Interfaces*, 2012.

[16] Nodira Khoussainova, YongChul Kwon, Magdalena Balazinska, and Dan Suciu. Snipsuggest: Context-aware autocompletion for sql. *Proceedings of the VLDB Endowment*, 4(1):22–33, 2010.

[17] Doris Jung-Lin Lee, Himel Dev, Huizi Hu, Hazem Elmeleegy, and Aditya G. Parameswaran. Storyboard: Navigating through data slices with distributional awareness.

[18] Doris Jung-Lin Lee, John Lee, Tarique Siddiqui, Jaewoo Kim, Karrie Karahalios, and Aditya G. Parameswaran. Accelerating scientific data exploration via visual query systems. *CoRR*, abs/1710.00763, 2017.

[19] Miro Mannino and Azza Abouzied. Expressive time series querying with hand-drawn scale-free sketches. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 388:1–388:13, New York, NY, USA, 2018. ACM.

[20] Alexandra Meliou, Wolfgang Gatterbauer, Joseph Y Halpern, Christoph Koch, Katherine F Moore, and Dan Suciu. Causality in databases. *IEEE Data Engineering Bulletin*, 33(EPFL-ARTICLE-165841):59–67, 2010.

[21] Mohebbi et al. Google correlate whitepaper. 2011.

[22] Kristi Morton, Magdalena Balazinska, Dan Grossman, and Jock Mackinlay. Support the Data Enthusiast: Challenges for Next-Generation Data-Analysis Systems. *Proceedings of the VLDB Endowment, Volume 7, pp. 453456, 2014*, 7:453–456, 2014.

[23] Arnab Nandi, Lilong Jiang, and Michael Mandel. Gestural query specification. *Proceedings of the VLDB Endowment*, 7(4):289–300, 2013.

[24] Christopher Olston and Ed Chi. ScentTrails: Integrating browsing and searching on the Web. *ACM Transactions on Computer Human Interaction TOCHI*, 10(3):177–197, 2003.

[25] E. Pierson, C. Simoiu, J. Overgoor, S. Corbett-Davies, V. Ramachandran, C. Phillips, and S. Goel. A large-scale analysis of racial disparities in police stops across the united states, 2017.

[26] Sudeepa Roy and Dan Suciu. A formal approach to finding explanations for database queries. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 1579–1590. ACM, 2014.

[27] Sunita Sarawagi, Rakesh Agrawal, and Nimrod Megiddo. Discovery-driven exploration of olap data cubes. EDBT '98, pages 168–182, Berlin, Heidelberg, 1998. Springer-Verlag.

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

[29] Ali Sarvghad, Melanie Tory, and Narges Mahyar. Visualizing Dimension Coverage to Support Exploratory Analysis. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):21–30, 2017.

[30] Vidya Setlur, Sarah E Battersby, Melanie Tory, Rich Gossweiler, and Angel X Chang. Eviza: A Natural Language Interface for Visual Analysis. *Proceedings of the 29th Annual Symposium on User Interface Software and Technology - UIST '16*, pages 365–377, 2016.

[31] Tarique Siddiqui, Albert Kim, John Lee, Karrie Karahalios, and Aditya Parameswaran. Effortless data exploration with zenvisage: An expressive and interactive visual analytics system. *Proceedings of the VLDB Endowment*, 10(4):457–468, November 2016.

[32] Tarique Siddiqui, Paul Luh, Zesheng Wang, Karrie Karahalios, and Aditya G. Parameswaran. Shapesearch: Flexible pattern-based querying of trend line visualizations. *Proceedings of the VLDB Endowment*, 2019.

[33] C. Stolte, D. Tang, and P. Hanrahan. Polaris: a system for query, analysis, and visualization of multidimensional relational databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–14, 2002.

[34] Manasi Vartak, Samuel Madden, and Aditya N Parmeswaran. SEEDB : Supporting Visual Analytics with Data-Driven Recommendations. *Proceedings of the VLDB Endowment, Volume 8, No. 13, 2015*, 2015.

[35] Manasi Vartak, Silu Huang, Tarique Siddiqui, Samuel Madden, and Aditya Parameswaran. Towards Visualization Recommendation Systems. *ACM SIGMOD Record*, 45(4):34–39, 2017.

[36] Martin Wattenberg. Sketching a graph to query a time-series database. In *CHI'01 Extended Abstracts on Human factors in Computing Systems*, pages 381–382. ACM, 2001.

[37] Leland Wilkinson. *The Grammar of Graphics (Statistics and Computing)*. Springer-Verlag, Berlin, Heidelberg, 2005.

[38] Wesley Willett, Jeffrey Heer, and Maneesh Agrawala. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1129–1136, 2007.

[39] Kanit Wongsuphasawat, Zening Qu, Dominik Moritz, Riley Chang, Felix Ouk, Anushka Anand, Jock Mackinlay, Bill Howe, and Jeffrey Heer. Voyager 2: Augmenting visual analysis with partial view specifications. In *ACM Human Factors in Computing Systems (CHI)*, 2017.

[40] Allison Woodruff and Michael Stonebraker. Supporting fine-grained data lineage in a database visualization environment. In *Proceedings of the Thirteenth International Conference on Data Engineering*, ICDE '97, pages 91–102, Washington, DC, USA, 1997. IEEE Computer Society.

[41] Eugene Wu and Samuel Madden. Scorpion: Explaining Away Outliers in Aggregate Queries. *Proceedings of the VLDB Endowment*, 6(8):553–564, 2013.

[42] Emanuel Zgraggen, Steven M Drucker, Danyel Fisher, and Robert DeLine. (s—qu)eries: Visual Regular Expressions for Querying and Exploring Event Sequences. *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, pages 2683–2692, 2015.

[43] Moshe M Zloof. Query by Example. *National Computer Conference*, pages 431–438, 1975.

Towards Quantifying Uncertainty in Data Analysis & Exploration

Yeounoh Chung[1], Sacha Servan-Schreiber[1], Emanuel Zgraggen[2], Tim Kraska[2]
[1]Brown University, Providence, RI, USA
[2]MIT CSAIL, Cambridge, MA, USA

Abstract

In the age big data, uncertainty in data constantly grows with its volume, variety, and velocity. Data is noisy, biased and error-prone; blindly applying even the most advanced data analysis techniques can easily mislead users to incorrect analytical conclusions. Also, compounding the problem of uncertain data is the uncertainty in data analysis and exploration. A typical end-to-end data analysis pipeline involves cleaning and processing the data, summarizing different characteristics and running more complex machine learning algorithms to extract interesting insights. The problem is that each step in the pipeline is error-prone and imperfect. From the input to the output, the uncertainty propagates and compounds. This paper discusses the challenges in dealing with various forms of uncertainty in data analysis and provides an overview of our work on Quantifying the Uncertainty in Data Exploration (QUDE), a toolset for safe and reliable data analysis.

1 Introduction

Tableau, scikit-learn, RapidMiner, Trifacta, Tamr, or Vizdom/IDEA are just a small collection of tools [2, 1, 18], which aim to make Data Science more accessible for everyone. However, democratizing Data Science also comes with a risk. For example, many of the "new" users these tools try to reach are not trained in statistics and thus, do not understand the nuances of the algorithms they use. Visual tools, like Tableau, make it possible to quickly test hundreds of hypotheses, and thus, it significantly increases the risk of finding false insights. Data integration tools, like Trifacta and Tamr, make it easier to integrate data, but might also hide data errors. Putting all these together in an end-to-end data analysis and exploration pipeline compounds and further increases the risks of uncertainty.

In this paper, we discuss several potential uncertainty in data analytics, and provide an overview of our work on Quantifying the Uncertainty in Data Exploration (QUDE): a new tool set to automatically quantify the different types of uncertainty/errors within data exploration pipelines. In one hand, there are the obvious types of uncertainty, such as outliers, which can be easily detected via simple visualizations (e.g., error bars, scatter plots). On the other hand, there are various types of uncertainty that are critical for reliable analysis results, but often overlooked. The goal of the QUDE project is to provide techniques for automatically detecting and quantifying such missed types of the uncertainty. This can help users without deep statistical or machine learning backgrounds to derive more safe and reliable data analytical conclusions.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

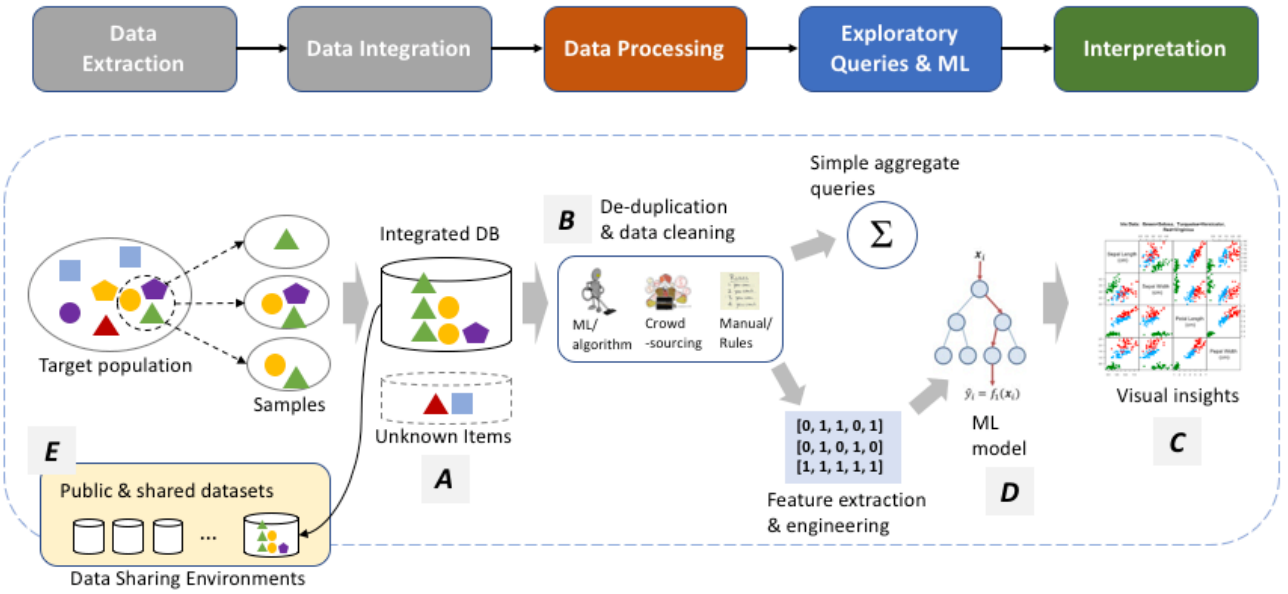


Figure 1: Data analysis & exploration workflow (top) and an example pipeline (bottom).

1.1 Types of Uncertainty in Data Analysis

A typical end-to-end data analysis pipeline involves collecting and cleaning data, summarizing different characteristics and running complex machine learning algorithms. At a high level, a data analysis and exploration workflow involves multiple stages illustrated in Figure 1. Namely, we need to extract high quality data from multiple data sources (or samples), clean the data to remove the inconsistency and merge the same entity in heterogeneous formats (i.e., de-duplication), apply data analysis techniques (e.g., simple aggregate queries or more complex machine learning algorithms) to extract useful information and extract interesting insights. Finally, we need to present the results in the right context for the interpretation.

While similar data analysis best practices can produce actionable insights and discoveries, *one should not take anything at face value*. The size and complexity, noise and incompleteness with big data not only impede the progress of the pipeline, but also make each step in the pipeline more error-prone. The uncertainty around the quality of the intermediate results propagates and compounds, making it even more difficult to validate the output results. Worse yet, the nature of data analysis and exploration tasks requires testing multiple hypotheses, applying different testing procedures with different combinations of data analysis techniques over the input data. The problem is that this "repeated attempts to find interesting facts, increases the chance to observe seemingly significant correlations by chance" [46], which is more formally known as Multiple Comparisons Problem (MCP) [28]. Obviously, many quality-related concerns exist in the entire pipeline from entity resolution problems up to Simpson-Paradox problems for aggregated results. In the following, we therefore highlight five types of uncertainty, which have been less explored in the literature and/or no automatic techniques exist to control the uncertainty type.

A. Missing Unknown Data Items [39, 15, 16]: Incompleteness of the data is one of the most common sources of uncertainty in practice. For instance, if unknown data items are missing (i.e., *we can't tell if the database is complete or not*) from the unknown target population, even a simple aggregate query result, like SUM, can be questionable. It is challenging to make sure that we have collected all the important data items to derive correct data analysis. On the contrary, the traditional survey and sampling methodologies work under a *closed-world* assumption, where there exist no unknown items.

B. Undetected Data Errors [14]: Complicating the challenges of incomplete data is also the quality of the collected data items. Real-world data is noisy and almost always comes with a variety of errors. Such

Table 1: More commonly-studied data analysis errors/uncertainty [12, 36, 26, 31, 35, 19, 20]. QUDE focuses on other types of uncertainty that are crucial for safe and reliable data analysis, but often overlooked in practice.

Data Extraction	Data Integration	Data processing	Exploratory Queries & ML	Interpretation
Sample selection bias	Entity resolution	Combination of error types	Model selection	Human involvement
Data source validity	Un-/semi-structured data	Data cleaning & coverage	Hyper-parameter tuning	Visualization selection
Disparate data sources	Data enrichment	Human involvement	Model bias & variance	Deceptive visualizations
		Missing information	Data under-/over-fitting	
			Feature engineering	
			Concept/distribution shift	

"dirty" data must be removed or corrected because errors can and will bias the results. There are many techniques to identify and repair the errors, but no single technique can guarantee a perfect error coverage. The challenge is that, one should use a number of orthogonal cleaning techniques or hire a lot of crowd-workers without knowing when to stop. Thus, we want to estimate *how many errors are still remaining in the data set*, without knowing the ground truth (a complete/perfect set of constraint rules or the true number of errors in the data set).

C. False Discovery [46]: Extracting insights from data requires repeated analysis and adjustment of hypotheses. With every new hypothesis and insight, the probability of encountering an interesting discovery by chance increases (also known as the MCP). Unfortunately, the problem is often overlooked in data analysis. In fact, many reported results, including published scientific findings, are false discoveries [27]. Thus, it is very important to control the MCP to ensure reliable data analysis and exploration.

D. Model Quality [13]: ML is one of the most popular tools for learning and making predictions on data. For its use, ensuring good ML model quality leads to more accurate and reliable data analysis results. The most common practice for model quality control is to consider various test performance metrics on separate validation data sets (e.g., cross-validation); however, the problem is that the overall performance metrics can fail to reflect the performance on smaller subsets of the data. At the same time, evaluating the model on all possible subsets of the data is prohibitively expensive, which is one of the key challenges in solving this uncertainty problem. Furthermore, missing unknown data items or sampling bias, in general, can also degrade the quality of the model. The challenge is that most ML/inference models perform badly on unseen instances, if similar examples are not learned during training.

E. Data Sharing Environments: When data is shared, a host of new problems increase uncertainty in data analysis. Namely, controlling false discoveries becomes much harder across several institutions or research groups given that the many hypotheses are posed against the shared data. A naïve solution would be to regulate the data sharing all together via a third-party service (*or don't share at all*). But this hinders scientific progress and is too costly to implement.

1.2 Our Goal and Contributions

As part of QUDE, we set out to quantify the uncertainty around the data analysis pipeline, which, in turn, should provide various measures to correct and validate the output results and discoveries. In this work, we provide an overview of our ongoing research, elaborating the uncertainty and its impact on the data analysis and exploration results, as well as the challenges associated with each case of uncertainty. Our initial prototype of QUDE focuses on the above five uncertainty types for two reasons: One, they are important for safe and reliable data analysis; Two, they are overlooked by the common data analysis and data wrangling systems [2, 1, 38, 3, 18]. Table 1 lists other types of errors or uncertainty that are more commonly considered in practice.

In the remainder of this paper, we discuss the above uncertainty cases in more detail and propose techniques to quantify and control the uncertainty (Sections 2, 3, 4, 5, 6); we conclude in Section 7 with some ideas on promising research directions as future work.

2 Uncertainty as Missing Unknown Data Items

First, we look at how uncertainty in a form of missing unknown data items (a.k.a., *unknown unknowns* [15, 16]) affects aggregate query results (e.g., AVG, COUNT, MIN/MAX), which are common in exploratory data analysis. It is challenging to make sure that we have collected all the important data items to derive correct data analysis, especially when we deal with real-world big data; there is always a chance that some items of unknown impacts are missing from the collected data set. To this end, we propose sound techniques to derive aggregate queries with the *open-world assumption* (the data set may or may not be complete).

2.1 An Illustrative Example

To demonstrate the impact of *unknown unknowns*, we pose a simple aggregate query to calculate the number of all employees in the U.S. tech industry, `SELECT SUM(employees) FROM us_tech_companies`, over a crowdsourced data set. We used techniques from [24] to design the crowdsourcing tasks on Amazon Mechanical Turk (AMT) to collect employee numbers from U.S. tech companies.[1] The data was manually cleaned before processing (e.g., entity resolution, removal of partial answers). Figure 2 shows the result.

The red line represents the ground truth (i.e., the total number of employees in the U.S. tech sector) for the query, whereas the grey line shows the result of the observed SUM query over time with the increasing number of received crowd-answers. As the ground-truth, we used the US tech sector employment report from the Pew Research Center [37]. The gap between the observed and the ground truth is due to the impact of the *unknown unknowns*, which gets smaller at a diminishing rate as more crowd-answers arrive.

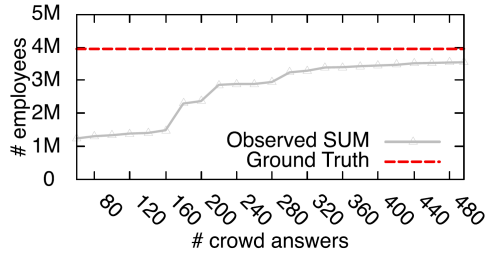


Figure 2: Employees in the U.S. tech sector

While the experiment was conducted in the context of crowdsourcing, the same behavior can be observed with other types of data sources, such as web pages.

2.2 Estimating The Impact of Unknown Unknowns

Estimating the impact of *unknown unknowns* for SUM queries is equivalent to solving two sub-problems: (1) estimating how many unique data items are missing (i.e., the *unknown unknowns* count estimate), and (2) estimating the attribute values of the missing data items (i.e., the *unknown unknowns* value estimate). The *naïve* estimator uses the $Chao92$ [9] species estimation technique to estimate the number of the missing data items, and *mean substitution* [36] to estimate the values of them.

Let $\phi_K = \sum_{r \in K} attr(r)$ be the current sum over the integrated database, then we can more formally define our *naïve* estimator for the impact of *unknown unknowns* as:

$$\Delta_{naive} = \underbrace{\frac{\phi_K}{c}}_{\text{Value estimate}} \cdot \underbrace{(\hat{N} - c)}_{\text{Count estimate}} \quad (1)$$

[1] More precisely, we only asked for companies with a presence in Silicon Valley, as we found it provides more accurate results.

$\hat{N}$ is the estimate of the number of unique data items in the ground truth D, and c is the number of unique entities in our integrated database K (thus, $\hat{N} - c$ is our estimate of the number of the unknown data items). ϕ_K/c is the average attribute value of all unique entities in our database K.

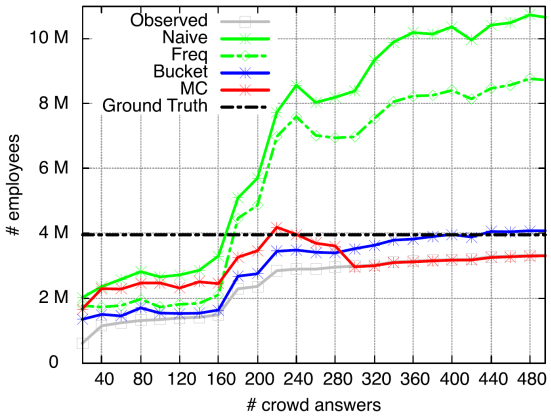


Figure 3: Employees in the U.S. tech sector estimation. While, *Naïve* approach heavily overestimates, *Bucket* estimator achieves the best results.

Figure 3 shows the results of the U.S. tech sector employment estimation. The *Naïve* approach heavily overestimates, since most observed companies are large (i.e., larger companies are more popular and likely to be sampled) and the value estimation is much higher than the true average number of employees. To account for this *publicity-value correlation*, we have proposed several estimators [16], and the *Bucket* estimator yielded the best results across different real-world data sets.

The idea of the *Bucket* estimator is to divide the attribute value range into smaller sub-ranges called buckets, and treat each bucket as a separate data set. We can then estimate the *impact of unknown unknowns* per bucket (e.g., large, medium, or small companies) and aggregate them to the overall effect:

$$\Delta_{bucket} = \sum_i \Delta(b_i) \tag{2}$$

Here $\Delta(b_i)$ refers to the estimate per bucket and both the *frequency* or *naïve* estimator could be used. The challenge with the *bucket* estimator is to determine the right size for each bucket. If the bucket size is too small, the bucket contains too few data items for any meaningful estimation. If the bucket size is too big, then the *publicity-value correlation* can still bias the estimate. Our *Bucket* estimator automatically splits the attribute value range to define buckets, which gives the most safe, conservative overall estimates.

The same techniques for *SUM*-aggregates can be applied to other aggregates for estimating the impact of *unknown unknowns*. For more details, as well as other proposed estimation techniques, we refer interested readers to our previous work [16].

3 Uncertainty as Undetected Data Errors

It is almost guaranteed that any real-world data sets contain some types of error (e.g., missing values, inconsistent records, duplicate entities). This is an important source of uncertainty in data analysis because those errors would almost surely corrupt the analysis results. Unfortunately, there has not been much work to measure the data quality or estimate the number of undetected/remaining data errors in a data set; the best practices in data cleaning basically employ a number of orthogonal data cleaning algorithms or crowd-source the task in a hope that the increased cleaning efforts would result in a perfect data set. As part of QUDE, we developed a new Data Quality Metric [14], which can guide the cleaning efforts.

3.1 An Illustrative Example

While this is a seemingly simple task, it is actually extremely challenging to define data quality without knowing the ground truth; previous works define data quality through counting the losses to gold standard data or violations of the constraint rules set forth by domain-specific heuristics and experts [44, 8, 11, 17, 32]. In practice, however, such ground truth data or rules are not readily available and are incomplete (i.e., there exists a "long tail" of errors). For instance, take a simple data cleaning task where we want to identify (and manually fix) malformed US home addresses in the database, shown in Figure 4. As in Guided Data Repair (GDR) [44], we

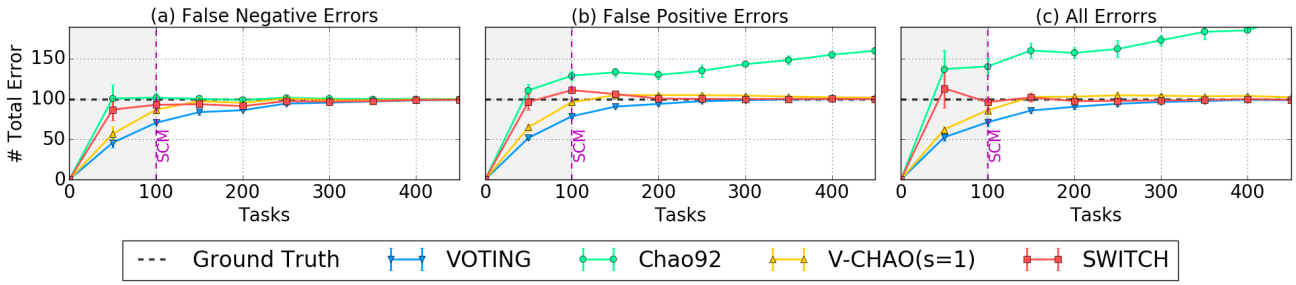


Figure 5: Total error estimates using the simulated datasets. The vanilla species estimation (*Chao92*) heavily overestimates in the presence of false positive errors; *SWITCH* is the most robust estimator against all error types.

might have a set of repair rules for missing values ($r1$, $r2$) and functional dependency violations ($r1$, $r3$, $r6$). However, the repair rules may not cover US state/city name misspellings ($r3$, $r4$) or wrong home addresses ($r5$, $r6$). Once errors are identified, a human can verify the proposed errors and automatic repairs. Similarly, as in CrowdER [42], we can run inexpensive heuristics to identify errors and ask crowd-workers to confirm. In both of these cases, the fallibility of the system in the form of false negative (e.g., "long tail" or missed errors) and false positive (e.g., even humans can make mistakes) errors is a big concern.

3.2 Data Quality Metric (DQM)

We want to design a statistical estimator to address both of the issues. That is, we need to estimate the number of remaining errors without knowing the ground truth in the presence of false negative and false positive errors. A simple approach is to extrapolate the number of errors from a small *"perfectly clean"* sample [43]: (1) we take a small sample, (2) perfectly clean it manually or with the crowd, and (3) extrapolate our findings to the entire data set. For example, if we found 10 new errors in a sample of 1000 records out of 1M records, we would assume that the total data set contains 10000 additional errors. However, this naïve approach presents a *chicken-and-egg* paradox. If we clean a very small sample of data, it may not be representative and thus will give an inaccurate extrapolation or estimates based off it. For larger samples, how can the analyst know that the sample itself is perfectly clean without a quality metric? In our work [14], we propose a *more robust and efficient way to estimate the number of all (eventually) detectable errors*.

	address	city	state	zip
r1:	15440 Southwest Mallard Drive Appartment # 101	Portland		97007
r2:	15440 SW Mallard Drive Apt # 102	Portland	OR	
r3:	12855 Southwest Dipper Lane Apartment # 101	Patland	OR	97007
r4:	289 Angell steet, unit 1H	Providence	RA	2912
r5:	Boston House, 239 S Indian River Dr	Fort Pierce	FL	34950
r6:	12345 ABCD street # EFGH	New York	NY	97007

Figure 4: Erroneous US home addresses: $r1$ and $r2$ contain missing values; $r3$ and $r4$ contain invalid city names and zip codes; $r1$, $r3$, and $r6$ violate a functional dependency ($zip \rightarrow city, state$); $r5$ is not a home address, and $r6$ is a fake home address in a valid format.

Interestingly, this problem is related to estimating the completeness of query results using species estimation techniques as first proposed in [40]. For the ease of exposition, let us assume that, as with most practical data cleaning tools, we rely on humans to verify the errors via crowd-sourcing (e.g., CrowdER [42]). Also, to overcome the human errors, we hire multiple workers to review each item. In this setting, we can think of our data quality problem as estimating the size of the set containing all distinct errors that we would discover upon adding more workers. The idea is to estimate the number of distinct records that will be marked erroneous if an infinite number of workers/resources are added, using species estimation techniques. Unfortunately, it turns out that false positives have a profound impact on the estimation quality of how many errors the data set contains.

Figure 5 shows the estimation results using different proposed techniques proposed. Note that the vanilla species estimation (*Chao92*) heavily overestimates in the presence of false positive errors. This is because

species estimators rely on the number of observed "rare" items as a proxy for the number of remaining species, and this number can be highly sensitive to a small number of false positive errors. To cope with the problem, we propose a more robust estimator (*SWITCH*) that estimates how the majority consensus on the items would flip.

Ideally, we want to estimate how many errors are still remaining in a data set. Instead, *SWITCH* estimates the total number of expected switches before the majority consensus converges to the ground truth (i.e., assuming workers are better than a random-guesser, the majority will eventually converge to the ground truth with enough votes). Switches act as a proxy to actual errors and, in many cases, might actually be more informative. However, since a record can switch from clean to dirty and then again from dirty to clean, it is not the same as the amount of dirty records or remaining errors in the data set. We estimate the total number of switches as with infinite workers, using the same $Chao92$ estimation technique; using this estimated quantity, we can adjust the current majority consensus to reach the ground truth:

$$majority(\mathcal{I}) + \xi^{+} - \xi^{-} \tag{3}$$

where positive switch ξ^{+} is defined as switches from the "clean" label to the "dirty" label and negative switch ξ^{-} as switches from "dirty" to "clean." It is important to note that this estimator is more robust against false positives, as it becomes less likely that, as the number of votes per item increases, a false positive will flip the consensus.

4 Uncertainty as False Discovery

Extracting insights from data requires repeated analysis and adjustment of hypotheses. With every new hypothesis and insight, the probability of encountering a chance correlation increases. This phenomenon is formally known as the multiple comparisons problem (MCP) and, when done intentionally, is often referred to as "p-hacking" [27] or "data dredging".

4.1 An Illustrative Example

Suppose we are looking for indicators in a census dataset that affects salary distribution. To examine factors such as "age" or "education", we set up the corresponding *null hypothesis* that states the proposed attribute has no correlation with the salary distribution. We then use a statistical test to infer the likelihood of observing a likewise spurious correlation under the null hypothesis. If this likelihood, commonly referred to as the p-value, is lower than the chosen significance level such as 0.05, then the null hypothesis is rejected, and the *alternative hypothesis* that the proposed attribute is correlated with salary is deemed statistically significant.

However, if we keep searching through different indicators in the dataset, we are almost guaranteed to find a statistically significant correlation. For example, choosing a significance level for each test of 0.05 means that statistically we have a 5% chance of falsely rejecting a given null hypothesis; even if the dataset contains completely random data, we would, on average, falsely discover a spurious correlation that passes our significance level after only 20 hypothesis tests.

4.2 Safe Visual Data Exploration

Visual data exploration tools such as Vizdom [18] or Tableau amplify this problem by allowing users to examine lots of visual "hypotheses" (e.g., comparing visualizations) in a short amount of time. In one of our experiments [45], where we used synthetic data sets with known ground truth labels, we found that by not accounting for all comparisons made during exploration, users are left with a high rate of false discoveries even if user-generated insights are followed up with statistical tests. Perhaps even more concerning is the increasing trend towards creating recommendation engines that propose interesting visualizations [41, 22] or automatically test for correlations [10]. Those systems are potentially checking thousands of hypotheses in just a few seconds. As a result,

it is almost guaranteed that such a system will find something "interesting" regardless of whether the observed phenomenon is statistically relevant or not [7].

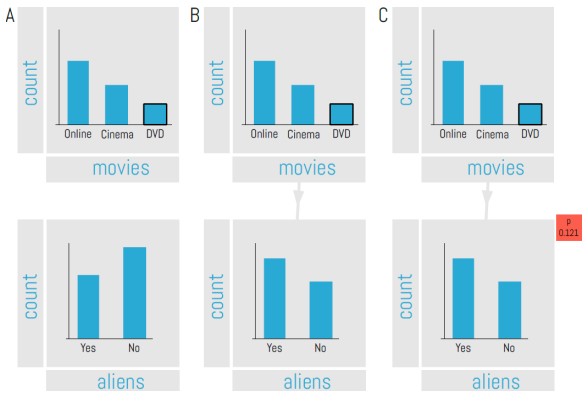


Figure 6: Example of a visualization network where users might be led to false discoveries without automatic hypothesis formulation. (A) two separate visualizations showing preferences for watching movies and how many people believe in alien existence; (B) the two visualizations combined where the bottom one shows proportions of belief in alien existence for only people who like to watch movies on DVD, displaying a noticeable difference compared to the overall population. (C) same visualizations as before but now with automatic hypothesis formulation turned on, highlighting that the observed effect is not statistically significant.

While there exists a variety of statistical techniques to control for the MCP [21, 5] they are not easily applicable in visual data exploration tools as they require knowledge about all the hypotheses being evaluated upfront, whereas in this context, the hypotheses are generated incrementally. To address this, we developed an MCP procedure [46] that allows specifying hypotheses incrementally. Furthermore, we show preliminary results on how such a procedure can be fully integrated into a data exploration system where visual comparisons are automatically tracked and controlled for [47]. Figure 6 shows an example of this. When we analyzed data from a survey on personal habits and opinions [7], we observed that the preference on watching films on DVD produced visually different proportions of belief in aliens, as shown in Figure 6 (A and B). Just by visually examining these charts, users often falsely assumed that people who prefer to watch movies on DVD are more prone to believe in aliens even though this effect is not statistically significant. When automatic testing and tracking is turned on, the system will try to formulate hypotheses for such cases (e.g., when users are comparing subsets against the global population), include them to the MCP control procedure and inform the user about the outcome of the test (Figure 6 C).

5 Uncertainty as Model Quality

ML is one of the most popular tools for uncovering hidden insights and patterns in data analysis. In fact, ensuring model quality leads to more accurate and reliable data analysis results. In this section, we discuss ML model quality as a form of uncertainty in data analysis. Namely, we look at a couple ML model quality issues that are often overlooked in practice. Model validation and quality assurance is an important component for QUDE.

5.1 An Illustrative Example

To ensure that a given model is performing well at a given task, people consider various test performance metrics (e.g., log loss, accuracy, recall, etc.). The problem, which is often overlooked is that the overall performance metrics can fail to reflect the performance on smaller subsets of the data. For example, we want to avoid a model that works well on average with the entire customer data, but fails with a female, teenage demographic in the U.S., especially, if it is one of the key market segments for the application. Here, we present an automated data slicing tool for model validation. The key challenge there is identifying a proper subset or data slice that is large, problematic and interpretable to the user; the search space is exponentially large with the number of features (and their value ranges).

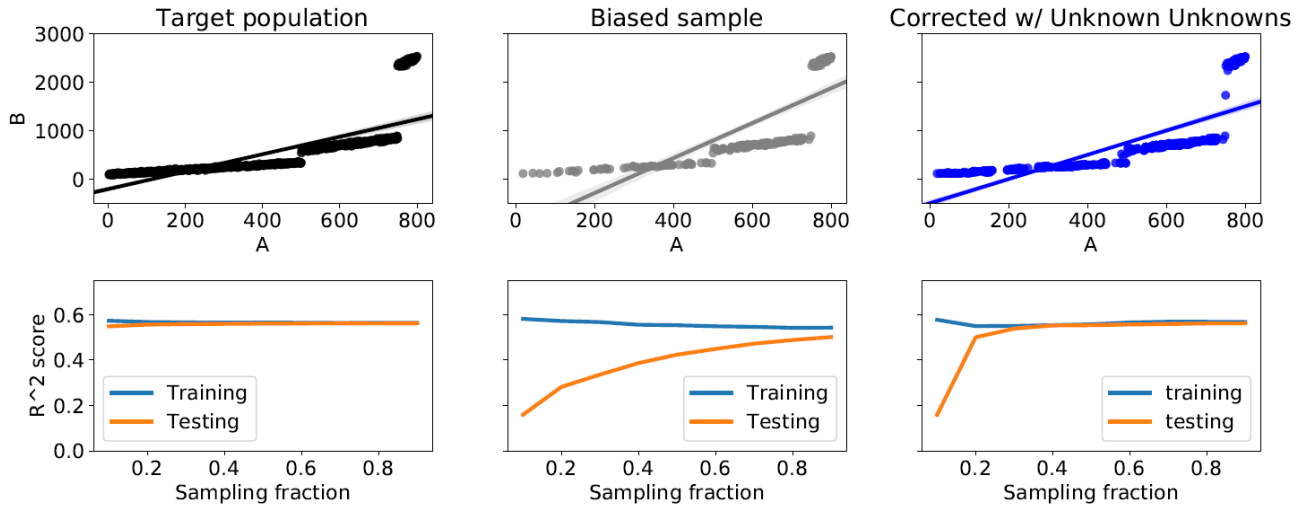


Figure 7: Ideally, we want the generalization gap between the training score and the testing score to be minimal (left); however, the model performs much worse if trained on a biased sample or fails to generalize to the actual testing data (middle). Accounting for *unknown unknowns* can improve the model generalizability (right).

5.2 Automated Data Slicing

While a well-known problem [33], current techniques to determine under-performing slices largely rely on domain experts to define important sub-populations (or at least specify a feature dimension to slice by) [30]. Unfortunately, ML practitioners do not necessarily have the domain expertise to know all important under-performing slices in advance, even after spending a significant amount of time exploring the data. In [13], we present an automated data slicing tool for model validation, called *Slice Finder*. The goal of *Slice Finder* is to identify a handful (e.g., top-K) of the largest problematic slices, that are also interepretable. Larger slices are preferable because they carry more examples, and thus, more impactful to model quality. On the contrary, debugging the model on a tiny slice would not mean much to the overall model quality. Plus, we want to bring the user's attention to the slices that are interpretable. For instance, country = US is more interpretable than country = US $\wedge$ age = 20-40 $\wedge$ zip = 12345, with a fewer number of common features. We find that the interepretability is a key for understanding the model quality problem. The resulting slices are presented via an interactive visualization front-end, which helps users to quickly browse through the slices.

5.3 Unknown Unknowns for ML

Another important aspect of ML model quality is generalizability, which measures how accurately an ML model can predict on new unseen examples. This is also important because ML-model-based analysis is often done to forecast or predict the future instances. Unseen examples during the training present a challenge to any inference model.

Figure 7 illustrates the problem. In the toy example, the target population is hidden (only used for testing), but the training data, which is a biased sample from the population, is missing some of the examples with smaller A values (e.g., smaller companies less likely to be sampled). The fitted regression model can still perform well on the training set, but will fail in testing. The wide gap between the training and the testing scores (middle) indicates this failure of model generalization. Now, by accounting for the *unknown unknowns* (e.g., injecting the generated unseen examples), we can improve the model generalizability. Of course, estimating the number, as well as the values of the unseen examples is not straight-forward.

6 Uncertainty in Data Sharing Environments

QUDE is much more challenging when data is shared or made public. Controlling false discoveries, discussed in Section 4, becomes more problematic when data is shared across institutions or research groups given the difficulty of establishing effective MCP control procedures in sharing environments. Even if just one member deviates from the exact testing protocol, uncertainty is immediately introduced into the results. Moreover, when data is made public, avoiding uncertainty as a result of the MCP (whether introduced intentionally or otherwise) becomes almost impossible given the inordinate amount of coordination and oversight required by all parties using the data. However, these issues do not impede the trend by industry and research institutions to make data publicly available making it imperative to create a method for effectively controlling the MCP in data sharing settings.

6.1 An Illustrative Example

Consider a publicly shared dataset such as MIMIC III [29] published by MIT. The dataset contains de-identified health data associated with $\approx 40,000$ critical care patients. The existing solution for controlling the MCP on such a dataset is to make use of a hold-out. In this case, MIT can release 30K patient records as an exploration dataset (EDS) and hold back 10K as a validation dataset (VDS). The EDS can then be used in arbitrary ways to find interesting insights from the data. However, before a result can be used in a publication, the hypothesis is tested for statistical significance over the VDS.

Unfortunately, there are several issues with such a solution. In order to use the VDS more than once, every hypothesis over the VDS has to be tracked and the MCP controlled. Hence, it becomes necessary for the data owner (e.g., MIT) to provide a "certification" service to validate results obtained over the EDS which is both a burden for the data owner as well as a potential source of bias. Researchers need to trust the data owner to correctly apply MCP control procedures and objectively evaluate their hypotheses.

6.2 Automated Result Certification

Ideally, a data owner publishes a dataset and goes offline (i.e., not have to interact with researchers any further) in order to both minimize the overhead imposed on the data owner as well as eliminate potential bias during the validation phase. An automated solution can be constructed using several cryptographic primitives and is based on the following observation: if all the p-values computed for a dataset are accounted for during the analysis phase, in addition to the order in which they were computed, it is possible to apply an incremental control procedure [46, 23] to control for uncertainty. The solution hinges on the use of *Fully Homomorphic Encryption* (FHE) which was first proposed by Gentry in 2009 [25]. FHE enables computation over encrypted data without revealing any information on the underlying data. Using FHE, any arithmetic operation (and thus any function) can be evaluated on ciphertexts using only the public key such that the result of the evaluation remains encrypted and unknown to the evaluator. If the data is encrypted by the data owner using FHE and the encrypted data made public, researchers are still able to use the encrypted data for analysis (e.g., compute statistical tests) and obtain encrypted results without interacting with the data owner. A researcher may then request the entity in possession of the decryption key (e.g., the data owner) to reveal the result computed locally over the encrypted data.

Consider, once more, the case of the MIMIC III dataset. The data owner (e.g., MIT) makes the encrypted version of the dataset publicly available. Researchers use the encrypted data to compute statistical tests and obtain encrypted p-values. The encrypted p-values are then sent to MIT which proceeds to decrypt and reveal each p-value on a publicly readable database. A p-value obtained in such a fashion can be audited by examining the sequence of test records stored in the database and applying an incremental control procedure. While still requiring participation of the data owner, such a solution is one step closer to the desired goal since it no longer requires the owner to verify hypotheses or otherwise interact with researchers.

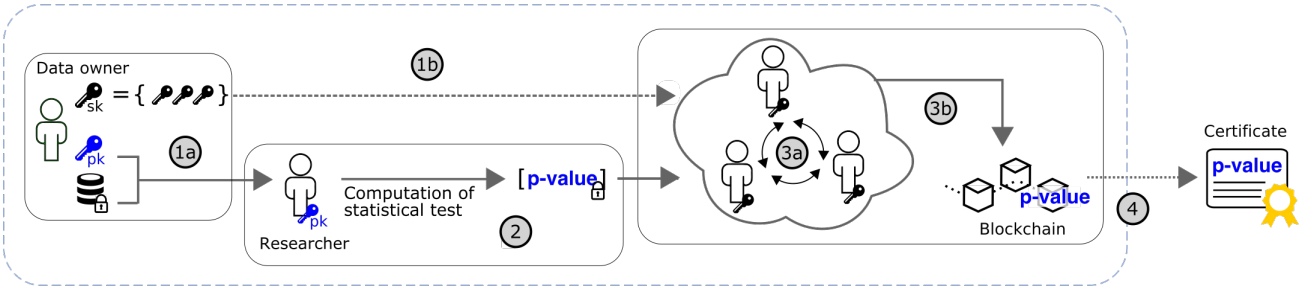


Figure 8: Overview of statistical test certification process: (1a) The data owner encrypts and publishes a dataset along with the public key. (1b) The data owner distributes the secret key to a set of parties that can collectively decrypt using threshold-FHE. (2) A researcher downloads the encrypted data along with the public key and computes a statistical test using FHE. (3a) To obtain the p-value in the clear, the researcher submits the encrypted result to the set of parties who then decrypt the p-value using a consensus protocol. (3b) The parties reveal the result by posting it on a public blockchain along with a timestamp. (4) A publication claiming a significant result can provide proof of valid testing procedures using the records stored on the blockchain.

The responsibilities of the data owner can be distributed to a set of parties (i.e., institutions, research groups, etc.) using threshold-FHE [4] which requires a majority of parties to "agree" on decrypting a result. In combination with a distributed ledger (e.g., a blockchain [34]), which guarantees immutability of recorded transactions, the sequence of tests can be tracked by recording each p-value at the time of decryption. Since the ledger is public and tamper-proof, it can be used as a mathematical proof of result validity. Figure 8 provides a high-level overview of the protocol.

7 Conclusion & Future Work

In this paper, we present several cases for the uncertainty in data analysis and exploration. First, we provide an overview of our work on quantifying the uncertainty as a form of unknown missing data items, undetected data errors in the data set. Next, we point out that any data-driven discoveries should be taken with care, because data analysis and exploration generally requires testing numerous hypotheses, increasing the chance of false discovery. Data analysis in a data sharing environment further complicates this issue of MCP. Finally, we discuss a couple model quality problems that can serve as a source of uncertainty in data analysis.

Our overarching goal is to quantify all types of uncertainty in data analysis and exploration, and in turn, provide measures to correct and validate the analysis results and discoveries. To this end, we plan on implementing the proposed solutions for ML model quality (Section 5) and data-sharing environments (Section 6). We have integrated some of the proposed solutions into an interactive human-in-the-loop data exploration and model building suite [6]. We are also interested in investigating other types of uncertainty (e.g., learning the right feature set for the task and the feature quality assurance) and how they interact with each other. It is interesting to understand the relationship among the different types of uncertainty as the uncertainty compounds over the pipeline.

References

[1] Lessons from next-generation data wrangling tools. https://www.oreilly.com/ideas/lessons-from-next-generation-data-wrangling-tools, 2015.

[2] The 6 components of open-source data science/ machine learning ecosystem; did python declare victory over r? https://www.kdnuggets.com/2018/06/ecosystem-data-science-python-victory.html, 2018.

[3] Z. Abedjan, X. Chu, D. Deng, R. C. Fernandez, I. F. Ilyas, M. Ouzzani, P. Papotti, M. Stonebraker, and N. Tang. Detecting data errors: Where are we and what needs to be done? *PVLDB*, 9(12):993–1004, 2016.

[4] G. Asharov, A. Jain, A. López-Alt, E. Tromer, V. Vaikuntanathan, and D. Wichs. Multiparty computation with low communication, computation and interaction via threshold fhe. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 483–501. Springer, 2012.

[5] Y. Benjamini and Y. Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society Series B (Methodological)*, 57(1):289–300, 1995.

[6] C. Binnig, B. Buratti, Y. Chung, C. Cousins, T. Kraska, Z. Shang, E. Upfal, R. C. Zeleznik, and E. Zgraggen. Towards interactive curation & automatic tuning of ml pipelines. In *DEEM@ SIGMOD*, pages 1–1, 2018.

[7] C. Binnig, L. De Stefani, T. Kraska, E. Upfal, E. Zgraggen, and Z. Zhao. Toward sustainable insights, or why polygamy is bad for you. In *CIDR*, 2017.

[8] P. Bohannon, W. Fan, M. Flaster, and R. Rastogi. A cost-based model and effective heuristic for repairing constraints by value modification. In *Proceedings of SIGMOD*, pages 143–154, 2005.

[9] A. Chao and S. Lee. Estimating the Number of Classes via Sample Coverage. *Journal of the American Statistical Association*, 87(417):210–217, 1992.

[10] 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, 2016.

[11] J. Chomicki and J. Marcinkowski. Minimal-change integrity maintenance using tuple deletions. *Information and Computation*, 197(1-2):90–121, 2005.

[12] X. Chu, I. F. Ilyas, S. Krishnan, and J. Wang. Data cleaning: Overview and emerging challenges. In *Proceedings of SIGMOD*, pages 2201–2206, 2016.

[13] Y. Chung, T. Kraska, S. E. Whang, and N. Polyzotis. Slice finder: Automated data slicing for model interpretability. *arXiv preprint arXiv:1807.06068*, 2018.

[14] Y. Chung, S. Krishnan, and T. Kraska. A data quality metric (dqm): how to estimate the number of undetected errors in data sets. *Proceedings of the VLDB Endowment*, 10(10):1094–1105, 2017.

[15] Y. Chung, M. L. Mortensen, C. Binnig, and T. Kraska. Estimating the impact of unknown unknowns on aggregate query results. In *Proceedings of ACM SIGMOD*, pages 861–876, 2016.

[16] Y. Chung, M. L. Mortensen, C. Binnig, and T. Kraska. Estimating the impact of unknown unknowns on aggregate query results. *TODS*, 43, 2018.

[17] G. Cong, W. Fan, F. Geerts, X. Jia, and S. Ma. Improving data quality: Consistency and accuracy. In *PVLDB*, pages 315–326, 2007.

[18] A. Crotty, A. Galakatos, E. Zgraggen, C. Binnig, and T. Kraska. Vizdom: interactive analytics through pen and touch. *Proceedings of the VLDB Endowment*, 8(12):2024–2027, 2015.

[19] Ç. Demiralp, P. J. Haas, S. Parthasarathy, and T. Pedapati. Foresight: Rapid data exploration through guideposts. *arXiv preprint arXiv:1709.10513*, 2017.

[20] X. L. Dong and D. Srivastava. Big data integration. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 1245–1248. IEEE, 2013.

[21] O. J. Dunn. Multiple comparisons among means. *Journal of the American Statistical Association*, 56(293):52–64, 1961.

[22] H. Ehsan, M. A. Sharaf, and P. K. Chrysanthis. Muve: Efficient multi-objective view recommendation for visual data exploration. In *ICDE*, pages 731–742, May 2016.

[23] D. P. Foster and R. A. Stine. α-investing: a procedure for sequential control of expected false discoveries. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 70(2):429–444, 2008.

[24] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin. Crowddb: Answering queries with crowdsourcing. In *Proceedings of ACM SIGMOD*, pages 61–72, 2011.

[25] C. Gentry. *A fully homomorphic encryption scheme*. Stanford University, 2009.

[26] I. Guyon and A. Elisseeff. An introduction to variable and feature selection. *Journal of machine learning research*, 3(Mar):1157–1182, 2003.

[27] M. L. Head, L. Holman, R. Lanfear, A. T. Kahn, and M. D. Jennions. The extent and consequences of p-hacking in science. *PLoS biology*, 13(3):e1002106, 2015.

[28] J. P. Ioannidis. Why most published research findings are false. *PLoS medicine*, 2(8):e124, 2005.

[29] A. E. Johnson, T. J. Pollard, L. Shen, H. L. Li-wei, M. Feng, M. Ghassemi, B. Moody, P. Szolovits, L. A. Celi, and R. G. Mark. Mimic-iii, a freely accessible critical care database. *Scientific data*, 3:160035, 2016.

[30] M. Kahng, D. Fang, and D. H. P. Chau. Visual exploration of machine learning results using data cube analysis. In *HILDA*, page 1. ACM, 2016.

[31] R. Kohavi et al. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.

[32] A. Lopatenko and L. Bravo. Efficient approximation algorithms for repairing inconsistent databases. In *Proceedings of ICDE*, pages 216–225, 2007.

[33] H. B. McMahan, G. Holt, D. Sculley, M. Young, D. Ebner, J. Grady, L. Nie, T. Phillips, E. Davydov, D. Golovin, et al. Ad click prediction: a view from the trenches. In *KDD*, pages 1222–1230, 2013.

[34] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. http://www.bitcoin.org/bitcoin.pdf.

[35] N. M. Nasrabadi. Pattern recognition and machine learning. *Journal of electronic imaging*, 16(4):049901, 2007.

[36] J. W. Osborne. *Best practices in data cleaning: A complete guide to everything you need to do before and after collecting your data*. Sage, 2012.

[37] Pew Research Center. How u.s. tech-sector jobs have grown, changed in 15 years. `http://pewrsr.ch/PtqZDA`, 2014. Accessed: 2015-07-08.

[38] M. Stonebraker, D. Bruckner, I. F. Ilyas, G. Beskales, M. Cherniack, S. B. Zdonik, A. Pagan, and S. Xu. Data curation at scale: The data tamer system. In *CIDR*, 2013.

[39] B. Trushkowsky, T. Kraska, M. J. Franklin, P. Sarkar, and V. Ramachandran. Crowdsourcing enumeration queries: Estimators and interfaces. In *to appear at IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2015.

[40] B. Trushkowsky, T. Kraska, and P. Sarkar. Answering enumeration queries with the crowd. *Commun. ACM*, 59(1):118–127, 2016.

[41] M. Vartak, S. Madden, A. G. Parameswaran, and N. Polyzotis. SEEDB: automatically generating query visualizations. *PVLDB*, 7(13):1581–1584, 2014.

[42] J. Wang, T. Kraska, M. J. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 5(11):1483–1494, 2012.

[43] J. Wang, S. Krishnan, M. J. Franklin, K. Goldberg, T. Kraska, and T. Milo. A sample-and-clean framework for fast and accurate query processing on dirty data. In *Proceedings of SIGMOD*, pages 469–480, 2014.

[44] M. Yakout, A. K. Elmagarmid, J. Neville, M. Ouzzani, and I. F. Ilyas. Guided data repair. *PVLDB*, 4(5):279–289, 2011.

[45] E. Zgraggen, Z. Zhao, R. Zeleznik, and T. Kraska. Investigating the effect of the multiple comparisons problem in visual analysis. In *CHI*, page 479. ACM, 2018.

[46] Z. Zhao, L. De Stefani, E. Zgraggen, C. Binnig, E. Upfal, and T. Kraska. Controlling false discoveries during interactive data exploration. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 527–540. ACM, 2017.

[47] Z. Zhao, E. Zgraggen, L. De Stefani, C. Binnig, E. Upfal, and T. Kraska. Safe visual data exploration. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1671–1674. ACM, 2017.

Query Perturbation Analysis: An Adventure of Database Researchers in Fact-Checking

Jun Yang
Duke University
junyang@cs.duke.edu

Pankaj K. Agarwal
Duke University
pankaj@cs.duke.edu

Sudeepa Roy
Duke University
sudeepa@cs.duke.edu

Brett Walenz
Duke University
bwalenz@cs.duke.edu

You Wu
Google Inc.
wuyou@google.com

Cong Yu
Google Inc.
congyu@google.com

Chengkai Li
University of Texas at Arlington
cli@cse.uta.edu

1 Introduction

Consider a database query with a number of parameters whose values can be specified by users. *Query perturbation analysis* examines how the result of this query varies in response to changes in its parameter values (while the database stays the same). Query perturbation analysis allows us to evaluate the sensitivity of a query's result to its parameters. For example, if we use queries over data to inform decisions or debates, we naturally would like to know whether the conclusions are "brittle" with respect to particular parameter choices. Moreover, by constructing a broader context formed by the results of a query over a large parameter space, query perturbation analysis allows us to further explore this context, such as searching for parameter settings that lead to robust decisions or convincing arguments.

An especially compelling application of query perturbation analysis is in *computational journalism* [10, 9]—specifically, how to fact-check claims derived from structured data automatically. Indeed, this application was what initially drew the authors of this paper to the problem of query perturbation analysis (and to coin this term in [40, 41]), as part of a long-term collaboration among a diverse group of researchers and practitioners from computer science, journalism, and public policy, across Duke University, University of Texas at Arlington, Stanford University, and Google. This paper provides an overview of our research on query perturbation analysis and its applications to journalism, and describes our collective experience and lessons learned—largely from the angle of database researchers.

The connection between fact-checking and perturbation analysis may not be immediately obvious. To start, we observe that in this age of data ubiquity, our media are saturated with claims of "facts" made from data, by politicians, pundits, corporations, and special interest groups alike. Based on data and often quantitative in nature, these claims have seemingly stronger credence in the eyes of the public. Indeed, we can verify their correctness by posing corresponding queries over the underlying data and ensuring that "the numbers check out." However, fact-checking goes beyond verifying correctness. Many claims can be correct yet still mislead. As an old saying goes, "if you torture the data long enough, it will confess to anything" [20]. Think of a claim—e.g., "adoptions went up 65 to 70 percent" when Giuliani "was the New York City mayor"[1]— as a query over data.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1] www.factcheck.org/2007/05/levitating-numbers/

The data (here, historical adoption numbers) may be pristine and the form of the query (comparing adoption numbers from two time periods) is completely innocuous. However, the choice of query parameters (comparing 1996–2001 and 1990–1995, which was not made explicit by the original claim) still controls what aspect or subset of the data to present in the claim, and can heavily influence the query result and hence the conclusion of the claim. Query perturbation analysis helps detect and counter such "cherry-picking" practices, because it puts the original claim into a larger context of perturbations for scrutiny; e.g., what if we compare the numbers from the mayor's first and second terms? In Section 2, we show how to apply perturbation analysis to this and other fact-checking tasks, as well as the complementary problem of finding leads from data while avoiding pitfalls.

Query perturbation analysis is perfectly suited to automation. Naively, we can evaluate the query with each possible parameter setting independently, so computation is embarrassingly parallel. In practice, however, brute-force solutions may not be acceptable because the number of possible query parameter settings can be daunting even if the database itself is small in size. In Section 3, we give an overview of techniques for efficient and scalable perturbation analysis. While there has been little prior work from the database community that directly addresses perturbation analysis, we note that some well-studied problems can be framed as perturbation analysis for specific forms of queries. On the other hand, even if we already understand very well how to process a query itself, its perturbation analysis often involves new technical challenges. There is a trade-off between the generality and efficiency of techniques for specific query forms: while we can usually do better with algorithms highly specialized for the occasion, there is a strong motivation to develop techniques and systems that can benefit any database queries. How to close the efficiency gap between general and specialized techniques remains an open challenge.

Interestingly, over the course of our research, we have discovered opportunities to transfer techniques for perturbation analysis to traditional database query processing (and back). Conceptually, we can write perturbation analysis of a query template q as a complex database query, starting with a subquery that enumerates all possible parameter settings for q, then "joining" it with a subquery that computes q for a given parameter setting (with parameters serving as join attributes), and finally applying some aggregation to analyze all results over the entire parameter space. Specifically, Section 3.3 describes some results we obtained exploring this connection, which include new optimizations for OLAP queries inspired by perturbation analysis. This pleasant surprise is a reminder of how pursuit of novel applications could lead to new advances for classic problems.

Last but not least, we reflect on our experience working on perturbation analysis and its applications to journalism in Section 4. We give a brief history of our collaboration with journalists, highlight a few interesting applications we built, and discuss the broader challenges we faced when dealing with the complexity of reality. We then conclude with an outlook for the future research on query perturbation analysis.

2 Modeling Fact-Checking and Lead-Finding

We use a couple of real-life examples to illustrate how to apply query perturbation analysis to a variety of fact-checking and lead-finding tasks. We shall focus on modeling these tasks conceptually for now, and leave the discussion of how to solve them efficiently to Section 3. Our first example is from the domain of US politics.

Example 1 (high correlation claims about voting records): *Evan Bayh, a Democrat from Indiana, ran for the US Senate in 2016. An ad aired during the 2016 elections attacked his records while serving in the Senate earlier, for voting with President Obama 96% of the time.*[2] *Here, the apparently high agreement percentage was used to invoke the impression that Bayh was hyperpartisan and a blind supporter of Obama.*

Ignoring a number of details for now, let us assume that we already have in our database the history of how each senator voted with Obama over time. Then, we can model the above claim as a SQL aggregation query about Bayh's history in particular. The query is parameterized by the time interval of comparison, although the

[2] `www.youtube.com/watch?v=vXEM64l9yfM`

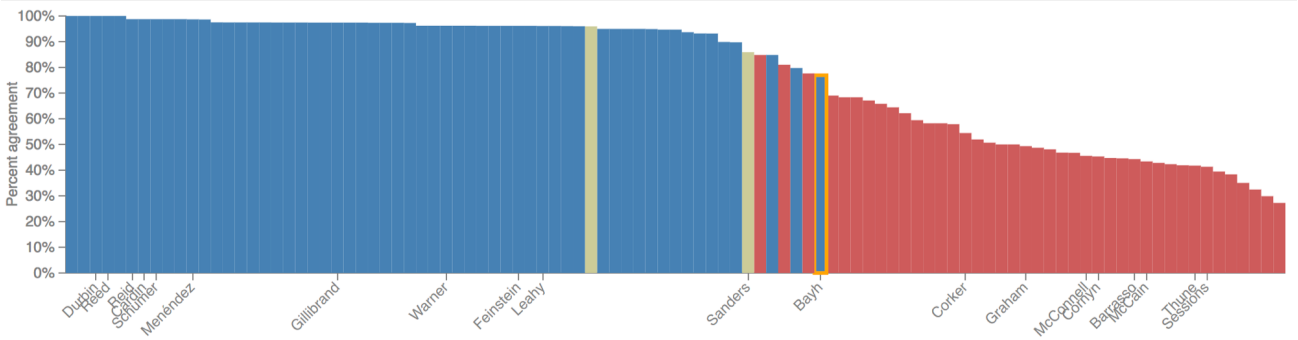


Figure 1: Visualization of how US senators voted with President Obama in 2009, generated by `congress.uclaimicheck.org`. Blue bars represent Democrats; red bars represent Republicans; two brass bars represent independents. The bar representing Evan Bayh is additionally outlined in orange. Data on votes in the Congress were from `GovTrack.us`, and data on the President's positions on these votes were from the *Congressional Quarterly*.

claim does not specify it. Such ambiguity is prevalent in political claims, and is often the very first challenge that fact-checkers must address, oftentimes with little to no help from those who made the claims. With perturbation analysis, however, we can "reverse-engineer" the query by searching for parameter settings that result in the claim's conclusion. In this example, we would look for plausible time intervals over which the aggregation query happens to return 96% (or something close).

As it turns out, the missing time interval in this query is the year 2010, the last year when Evan served in the Senate. The number 96% "checks out." However, as motivated in Section 1, a correct claim can still mislead. First, if we perturb the time interval slightly, to the year before, Bayh would have agreed with Obama only 77% of the time. Considering 2009–2010 would have resulted in 85%, still nowhere as impressive as 96%. In other words, perturbation analysis can reveal when a claim is not robust*, i.e., even small parameter changes can weaken its conclusion significantly.*

We can additionally or alternatively parameterize the same query by the senator we are comparing with Obama. Perturbation in this dimension allows us to see where Bayh stands in a larger context of all senators. Figure 1 (a screenshot from a website we built for the 2016 US elections) visualizes the result of this analysis. We see that in 2009, Bayh was in fact the Democrat who agreed least with Obama, and even less than three Republicans. For a Democrat, 77% agreement with Obama was really low, and 96% was not that unusual. In other words, perturbation analysis can reveal when a claim is not unique*, i.e., it is easy to make similar claims for many other entities besides the one being highlighted.* ■

Claims like the one above have been used repeatedly in political campaigns over the years, by both sides of the political aisle, for both support and attack. For instance, in [40], we used an example from the 2010 elections where an ad supporting Jim Marshall, a Democratic incumbent from Georgia, claimed that he "voted the same as Republican leaders 65 percent of the time." This ad was actually in response to an earlier ad attacking Marshall for voting with Nancy Pelosi (a notable Democrat who was the Speaker of the House at that time) "almost 90 percent of the time."

We should note that, by our observation, human fact-checkers do intuitively perform perturbation analysis, albeit manually (and perhaps instinctively rather than explicitly)—they often put a given claim in a larger context to probe its weaknesses and look for counterarguments. In this sense, perturbation analysis is a natural formalization of human intuition into a computational framework. This formalization brings a number of important new benefits, however. First, we can now elucidate and quantify various intuitive measures of claim quality, such as *robustness* and *uniqueness* motivated in Example 1. Second, we can formulate a variety of fact-checking tasks as computational problems. We have already mentioned in Example 1 the task of *reverse-engineering* claims to remove ambiguity. Another example is the task of *finding counterarguments* to claims. Intuitively, for a claim that is not robust, effective counterarguments would be small parameter changes that significantly weaken the

original claim's conclusion; for a claim that is not unique, an effective counterargument would be a collection of perturbations with similar or stronger conclusions in order to show that the original claim is not special. These tasks can be formulated as search or summarization problems in the space of perturbations. For more details about our framework, see [40, 41].

It is also natural to turn the problem of fact-checking around, and ask how we can find high-quality "leads" from data—claims that are not only interesting, but also robust and unique so they are not easily countered. To illustrate, we use an example about a lighter subject, sports journalism.[3]

Example 2 (one-of-the-few claims in sports): *Sports reporting and commentaries are often inundated with claims involving statistics that supposedly show how impressive a player or performance is. (For a good laugh, listen to the late Frank Deford's opinion on the NPR's Morning Edition.*[4]*) Consider, for example, the claim that "only 10 players in the NBA history had more points, more rebounds, and more assists per game than Sam Lacey in their career" [43]. Sam Lacey was indeed a great basketball player, but most fans probably would not rank him among the most legendary players as the impressive-sounding "only 10" part of the claim would suggest.*

Using perturbation analysis, we can model this claim as a counting query, parameterized by the player, which tallies the number of players who dominate the given players on all three attributes: points, rebounds, and assists per game. By perturbing the player of interest and checking whether the resulting count is 10 or less, we get to see for how many other players we can make the same or even stronger claims. For this example, it turns out that we do so for more than a hundred players, so the original claim about Sam Lacey is not unique.

Thus, to find unique claims, we would consider all possible parameter settings (players) as candidates. Given a candidate player o, suppose the count of players dominating o is k. To decide whether o is unique, we would count τ, the number of parameter settings that result in the same or stronger claims, i.e., the number of players dominated by k or fewer other players. If τ is small, then o is unique. In other words, instead of testing k, which is an unreliable indicator of a claim's interestingness, we test τ, which measures uniqueness under the perturbation analysis framework.

Interestingly, the uniqueness measure naturally penalizes claims of this form for comparing too many attributes. As dimensionality grows, dominance in all attributes becomes increasingly rare among players, so it becomes easier to make claims with impressively low dominance count k. But at the same time, τ would grow, so checking τ naturally protects us from making misleading claims that are not unique. ■

We have done a more in-depth study of how to mine these so-called "one-of-the-few" claims from data in [39]. This earlier work was not presented in the perturbation analysis framework, but the connection, especially to the uniqueness measure of claim quality, should be obvious. This work also allowed claims to compare different subsets of attributes. Although SQL queries traditionally would not consider the attributes they reference as "parameters," we can view changes to the referenced attributes as a general form of query perturbation, and our framework can be extended to accommodate perturbations in not only parameter values but also query forms.

So far, our description of query perturbation analysis has been mostly informal. Before proceeding to the discussion on computational techniques in the next section, we first lay out some formal definitions.

Definition 1 (Query Perturbation Analysis [37]): *Consider a (fixed) database instance $\mathcal{D}$, and a parameterized* query template *q over $\mathcal{D}$ with parameter settings drawn from a* parameter space *P. Let $q(p)$ denote the result of evaluating q with parameter setting $p \in P$ (over $\mathcal{D}$; omitted for brevity). Let $\mathcal{R} = \{(p, q(p)) \mid p \in P\}$ denote the collection of results (*perturbations*) obtained by evaluating q for all possible parameter settings. A*

[3] For an even lighter subject (though less related to journalism), we demonstrated at *SIGMOD* 2014 a system called *iCheck* that made interesting claims about the publication records of database researchers [43] (it also covered other less frivolous domains, of course). For example, *iCheck* found that one author of this paper had a great year of publications in *SIGMOD*, *VLDB*, and *ICDE*—only 8 other researchers had ever managed to publish as much as (or more than) him in one year in every one of these three venues. Then, *iCheck* turned around and noted that the same claim could be made for 89 other researchers (i.e., each of them also had a great year with publication records matched or dominated by no more than 8 others)!

[4] `www.npr.org/templates/story/story.php?storyId=98016313`

query perturbation analysis problem *is specified by a* 4*-tuple* $(\mathcal{D}, P, q, \chi)$*, where* χ *is a* post-processing function *that computes from* $\mathcal{R}$ *the final* answer set *for the analysis.* ■

As an example, we formalize the fact-checking task of *evaluating claim robustness* as follows. Let p_0 denote the parameter setting used by the claim we would like to check. To spell out χ, we need two helper functions:

- The *parameter sensibility function* $\mathsf{SP}(p, p_0)$, where $p \in P$, measures how "sensible" a perturbed parameter setting is with respect to the original. Not all perturbations are equally useful to investigation: we would only assign non-zero sensibility to perturbations that are relevant to the context of the claim, and we assign higher sensibilities to those that are more "natural" (e.g., using "2 years" in a statement is more natural than "22 months").

- The *result strength function* $\mathsf{SR}(q(p), q(p_0))$ measures how much the result of the perturbation deviates from the original. A negative $\mathsf{SR}(q(p), q(p_0))$ means the perturbation weakens the original claim.

Then, we can assess the robustness of the claim using $\chi(\mathcal{R}) = \sum_{(p,q(p))\in\mathcal{R}} \mathsf{SP}(p, p_0)\cdot(\min\{0, \mathsf{SR}(q(p), q(p_0))\})^2$. One way of interpreting this definition is to picture yourself as a "random fact-checker," who randomly perturbs the parameter setting according to sensibility—higher $\mathsf{SP}(p, p_0)$ means p is more likely to be chosen. Here, $\chi(\mathcal{R})$ computes the mean squared deviation from the original claim (considering only perturbations that weaken it) as observed by the random fact-checker—a high $\chi(\mathcal{R})$ means the claim is not robust.

As another example, consider a lead-finding task: *mining one-of-the-few claims with high uniqueness* (recall Example 2). We can generally define the uniqueness of a claim with parameter setting p_0 by computing $\frac{1}{|P|}\sum_{p\in P} \mathbf{1}(\mathsf{SR}(q(p), q(p_0)) < 0)$, i.e., the fraction of parameter settings for which the claim would be weaker than that for p_0. In this case, P is the set of objects of interest, $q(p)$ computes the number of objects dominating $p \in P$ on the given set of attributes for comparison, and $\mathsf{SR}(q(p), q(p_0)) = q(p_0) - q(p)$. Hence, to find claims with uniqueness at least $1 - \frac{\tau}{|P|}$, we use $\chi(\mathcal{R}) = \{(p, q(p)) \mid \tau \geq \sum_{(p',q(p'))\in\mathcal{R}} \mathbf{1}(q(p') \leq q(p))\}$.

We refer interested readers to [41] for more details as well as examples of formulating other fact-checking and lead-finding tasks in this framework. Besides the problems above, one notable practical issue, when finding counterarguments and mining leads, is that there may be too many candidate perturbations to return. One could simply return those that score the highest by some quality measure, but they may be too similar to each other, and some may be outliers and not representative of their neighboring perturbations. Therefore, in [42], we studied the problem of how to choose a small, diverse set of high-quality representatives from the space of perturbations. In general, much research is still needed on effectively analyzing and summarizing the context provided by the space of perturbations, which can require more complex post-processing functions than database queries.

3 Computational Techniques for Perturbation Analysis

A naive way of performing perturbation analysis is to first generate $\mathcal{R}$ by computing $q(p)$ for each possible parameter setting $p \in P$, and then evaluate $\chi(\mathcal{R})$. As mentioned in Section 1, this method is not feasible if P is large, especially when it involves multiple dimensions. In this section, we overview some techniques for more efficient perturbation analysis.

3.1 Query-Specific Methods

Interestingly, a number of well-studied problems in databases can be cast as specific query perturbation analysis tasks, so for these tasks, we can apply known techniques. For example, consider *frequent itemset mining* [6]. Let query template q, parameterized by an itemset T, count the number of transactions that contain all items in T. Then, finding all frequent itemsets is equivalent to the perturbation analysis task of selecting all parameter settings T for which $q(T)$ is above a threshold. For another example, recall Example 2 but consider instead

the simpler lead-finding task of identifying all players who are dominated by no more than k players. If we represent each player as a point in a space whose dimensions are the attributes being compared, then this task essentially amounts to computing the *k-skyband* [27] (a concept that generalizes *skyline* [7]) for a set of points in a high-dimensional space.

Much of perturbation analysis is not covered by existing work, however. For example, to find truly robust and unique claims, we often introduce criteria in the post-processing function χ that require different algorithms or adaptations of techniques from existing work. For instance, as motivated in Example 2, claims about players with low dominance counts are not necessarily interesting; rather, we should look for unique claims—ones that cannot be made for too many players. Hence, the problem shifts from that of computing the k-skyband for a given k, to that of computing the k-skyband for the maximum k such that the k-skyband still contains no more than τ points. Techniques for this new problem setting had to be developed [39].

We observe that practical applications of perturbation analysis often add some element of optimization in the post-processing function χ. Even if efficient evaluation of query template q has been thoroughly studied, the combination of q and an "optimizing" χ leads to interesting new problems, as the following example illustrates.

Example 3 (vote correlation claims and convex hulls; adapted from [41]): *Recall Example 1. Suppose we want to refute the claim about Bayh agreeing too much with Obama with a counterargument showing that the claim is not robust. We can search for this counterargument with an "optimizing" χ. Specifically, consider perturbing the time interval being compared by the claim. We would like to find the interval that minimizes the percentage agreement, subject to some constraints such as length and scope (we omit the details here).*

Putting aside χ for the moment, the query template q in this case is simply range aggregation. The standard technique of prefix sums *allows us to answer q for any query interval in constant time: by precomputing and remembering the running count of agreements over time, we can find the number of agreements over any interval by taking the difference between the values of the running count at the two interval endpoints. As Figure 2 shows, if we plot the running count over time, the percentage agreement over an interval is simply the slope of the line connecting the two endpoints.*

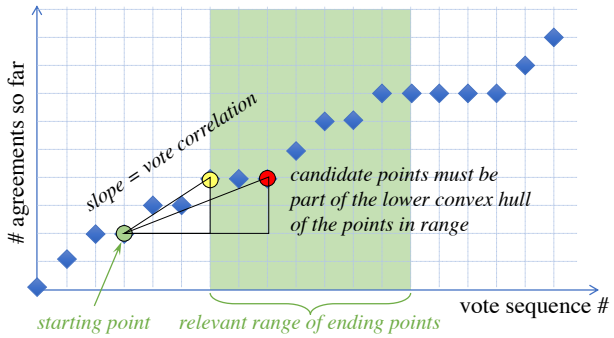


Figure 2: Minimizing percentage agreement by considering the convex hull of points formed by the running count of agreements over time.

However, solving the constrained optimization problem efficiently requires some new insights. Instead of considering all intervals, our technique in [41] aims at quickly finding the optimal interval among those with the same starting point (without explicitly examining all such intervals), and then choosing the best among all possible starting points. Given a starting point, we can determine the relevant range of ending points (as dictated by any given constraints). The key to speeding up the search for the optimal ending point in this range is a geometric intuition illustrated in Figure 2: any candidate must be part of the lower convex hull of the points in the range; there is no need to search the other points. Thus, the problem reduces to indexing for range convex hull queries (see [41] for details). ■

There are many other examples where considering perturbation analysis for well-studied query types breathes new life into old problems. While it is both interesting and useful to develop highly specialized solutions for specific query templates, they require a lot of human expertise and effort. We may not have such a luxury when so many queries and application domains could benefit from perturbation analysis. Thus, it is also important to devise more generic methods—which we discuss next.

3.2 Generic Methods

Generic methods aimed at speeding up the computation of $\mathcal{R}$ include *parallelization* (i.e., evaluating all perturbations in parallel); *grouping* (i.e., evaluating some perturbations together efficiently as a group); and *memoization* (i.e., caching the result of a piece of computation and reusing it later). The classic database problem of *MQO* (*multiple-query optimization* [29]) is closely related, although it traditionally targets very different query workloads. Perturbation analysis typically involves far more queries than MQO, but all of them have the same form and differ only in their parameter settings; MQO makes no such assumption and hence focuses more on finding common subqueries to share processing. Another key difference is that perturbation analysis has a post-processing function χ, which opens up the possibility of "pushing down" selection criteria from χ down into the computation of $\mathcal{R}$. These additional selection criteria in turn enable more *pruning* (i.e., using cached results for one set of inputs to help discard unpromising parameter settings without full evaluation). We illustrate the use of memoization and pruning with a simple example below.

Example 4 (memoization and pruning for finding one-of-the-few claims; adapted from [37]): *Recall Example 2. Suppose our task now is to find all claims of the form "player A's performance in stats X and Y during season S is dominated no more than* 10 *times." A baseline approach would be to process one player-season pair at a time, looking up the performance record (say (x, y)), counting the number of records dominating (x, y), and then checking whether this count is no more than* 10*. With memoization, we remember the outcome of this check together with the input—importantly, (x, y), instead of the player-season pair. Later, if we encounter another player-season pair that happens to yield the same (x, y), we can reuse the result of the check. How often does this simple memoization method help? The answer depends on how many distinct (x, y) pairs there are versus player-season pairs. If the number of players is large while stats have relatively smaller active domains, memoization can produce great savings—more than 90% reduction in the number of counting queries for some practical scenarios tested in [37].*

We can further improve the above approach with pruning. Once we find that a performance record (x, y) has a dominance count of c, we remember (x, y, c). Upon encountering a previously unseen record (x', y'), instead of immediately computing its dominance count, we first check the entries we remembered to see if we find some entry $(x^\star, y^\star, c^\star)$ where $(x^\star, y^\star)$ dominates (x', y') and $c^\star \geq k$. If yes, then we know, by transitivity of dominance, that (x', y') must have a dominance count greater than k and therefore can be discarded without further processing. ■

Note that even after applying memoization and pruning, as in the example above, the resulting procedure still may not be as efficient as a specialized method, e.g., a handcrafted k-skyband algorithm. However, the advantages of generic methods are their broader applicability and the possibility for automatic optimization, which significantly reduces development costs. How to close the efficiency gap between the general and specialized solutions is an ongoing challenge, which requires continuing to develop not only new generic methods but also better automatic optimization—a topic that we will come back to in Section 3.4.

One issue that we glossed over in Example 4 is how to come up with the pruning logic in the first place. If we must specify this logic by hand for every perturbation analysis task, some appeal of the generality of pruning would be lost. Fortunately, as we shall see in Section 3.3, sometimes we can derive the pruning logic automatically from a specification of the perturbation analysis task (e.g., for Example 4). However, for more complex pruning opportunities (see [37] for examples), identifying them automatically remains an open challenge.

Another suite of generic methods for perturbation analysis aim at exploiting properties of the post-processing function χ that commonly arise in applications. First, for many fact-checking and lead-finding tasks, χ uses a parameter sensibility function SP (introduced at the end of Section 2), which intuitively weighs some perturbations more than others. Instead of considering perturbations in some arbitrary order, a good heuristic would be to consider them in the decreasing order of sensibility, and stop when we are sure that remaining perturbations will no longer contribute to the answer because of low sensibility. Second, consider the case where χ involves

searching for the "best" parameter settings within a region of the parameter space defined by constraints. While optimization over the entire region may seem complex, sometimes we can divide the region into "zones" within which we know how to solve the optimization problem more efficiently (e.g., as in Example 3). In [40, 41], we developed various "meta" algorithms implementing these high-level generic methods, with pluggable low-level building blocks that can be customized for the occasion.

An exciting direction that we have been pursuing recently is the use of approximation methods for perturbation analysis. Approximation is effective for taming computational complexity when aggregating or optimizing over a huge, high-dimensional parameter space, and in most practical scenarios, we can tolerate some errors. The most straightforward method would be uniform random sampling, but alternative methods often work better in our context: e.g., *importance sampling* [32] based on the parameter sensibility function SP, and *coreset sampling* [4] for certain types of post-processing functions. We also borrow a number of ideas from machine learning. Acquiring each sample involves a potentially expensive query, but we can learn a model predicting the contribution of a parameter setting to the result of the analysis. This learned model can be used to inform sampling design, and techniques from *active learning* [30] and *quantification learning* [15] are also applicable. We hope to make our results on this front available soon.

3.3 Connections to Traditional Query Processing

As mentioned in Section 1, we can think of the perturbation analysis of a query template q as a complex query, with one part enumerating all possible parameter settings and the rest computing q for each parameter setting. If the post-processing function χ is not too complicated, it may be expressed as additional aggregation and/or selection. For instance, the two examples at the beginning of Section 3.1, mining frequent itemsets and finding k-skyband, can be written in SQL as follows (for simplicity, consider only 2-itemsets and 2-d skyband):

```
-- Basket(bid, item) stores one item per row
SELECT i1.item, i2.item
FROM Basket i1, Basket i2
WHERE i1.bid = i2.bid
GROUP BY i1.item, i2.item
HAVING COUNT(*) >= s;
```

```
-- Object(id, x, y) stores one object per row
SELECT L.id, COUNT(*)
FROM Object L, Object R
WHERE L.x<=R.x AND L.y<=R.y AND (L.x<R.x OR L.y<R.y)
GROUP BY L.id
HAVING COUNT(*) <= k;
```

These queries are both joins followed by grouping and further selection based on group sizes, which are examples of so-called *iceberg queries* [13] in OLAP. There has been plenty of work on computing frequent itemsets, skybands, and iceberg queries. Interestingly, their techniques have been developed largely disjointly: the techniques for the first two problems are quite specific, while work on iceberg queries does not typically address the joins that happen before grouping and aggregation. Perturbation analysis offers us an opportunity to reexamine these traditional query processing problems in a single framework.

A natural idea to try is applying the generic methods for perturbation analysis to iceberg query processing, which we did in [36]. To apply memoization and pruning, we devise a physical join operator called *NLJP*, or *nested-loop join with pruning*. NLJP operates as a nested loop, evaluating its inner(-loop) query with the join attribute values of each tuple produced by its outer(-loop) query. NLJP caches the results of the inner query by its input join attribute values, and for each new tuple from the outer query, evaluates a pruning predicate to determine whether it can safely skip the inner query evaluation. The cache also enables memoization.

Another style of pruning, which generalizes the *A-priori* technique for frequent itemset mining, works by first applying the HAVING constraint to an input table before it is joined. For example, for the query computing frequent 2-itemset above, we can replace each instance of Basket by

```
(SELECT * FROM Basket WHERE item IN
 (SELECT item FROM Basket GROUP BY item HAVING COUNT(*) >= s))
```

and drastically reduce work in subsequent processing. This optimization can be achieved by query rewriting.

The key technical challenge we tackled in [36] is how to identify and apply memoization and pruning techniques automatically, without relying on any manual hints. To this end, we have developed formal conditions for the applicability of these techniques. Interestingly, both NLJP- and A-priori-style pruning are underlined by query *monotonicity* properties, where we can infer useful relationships between query results when input tables have containment relationships. We analyze SQL queries with the knowledge of the database constraints to detect optimization opportunities. To apply them, we rewrite the query and/or insert NLJP operators into the query plan. Notably, we can deduce pruning predicates used by NLJP automatically, for query conditions involving linear equalities and inequalities, using quantifier and variable elimination methods [21].

Our implementation in PostgreSQL [1] shows substantial performance improvements over existing database systems for complex iceberg queries [36]—a nice example of how insights from perturbation analysis help with classic query processing. Beyond [36], we are actively applying the approximation methods discussed in Section 3.2 to query processing, to reduce the cost of evaluating expensive subqueries or user-defined predicates.

What makes the connection between perturbation analysis and traditional query processing doubly exciting is that it has also led to new breakthroughs for perturbation analysis. With the techniques developed in [36], we can automatically identify and apply memoization and pruning techniques for a fairly broad class of perturbation analysis tasks expressible as SQL iceberg queries. Another interesting follow-up problem is how to apply A-priori-style pruning techniques to perturbation analysis, especially when we generalize perturbations to not only parameter values but also query forms—this pruning style is particularly powerful across changes to the "dimensions" involved in the query.

3.4 System Support

We have built a system called *Perada* [37] to support perturbation analysis of SQL queries. We want this system to be general (supporting any query template expressed in SQL), scalable (capable of processing a large number of perturbations for time-sensitive fact-checking and lead-finding tasks), and easy to use (allowing developers to code new analyses quickly, without burdening them with low-level implementation and tuning). As mentioned earlier, a key challenge we face is the trade-off between generality and efficiency. The approach we have taken with Perada is to support a suite of generic methods including parallelization, grouping, memoization, and pruning, as discussed in Section 3.2, but rely on developers to specify how to apply these methods to the task at hand through a flexible, high-level API. Perada takes care of parallelization and hides the complexity of concurrency and failures from developers.

Another important feature of Perada is automatic tuning. While Perada relies on developers to specify parallelization, memoization, and pruning opportunities, it automatically makes various tuning decisions critical to performance: e.g., whether the benefit of memoization and pruning outweighs their overhead, or how to strike the balance between parallelism and serialism (the latter enables memoization and pruning). These decisions are difficult for developers to make, as the various methods interact with each other in subtle ways, and their effectiveness generally depends on factors such as data characteristics. We use an example to illustrate this point.

Example 5 (factors influencing pruning effectiveness; adapted from [37]): *Consider the pruning method in Example 4 for finding one-of-the-few claims. We apply it to a dataset about the Major League Baseball that records player performances in each season in terms of more than twenty stats. First, suppose we are interested in three stats: hits, home runs, and stolen bases, and the dominance count threshold for reporting a claim is* $k \leq 50$*. Figure 3 (ignore the dashed line plot for now) shows the pruning rate over the course of examining all player-season pairs sequentially in random order. We see that the pruning rate picks up rapidly at the beginning of execution, and converges to a very high percentage. Intuitively, as we see more perturbations, we build up a better picture of the "decision boundary" separating perturbations inside and outside the answer set, eventually allowing us to prune most perturbations lying on one side of the decision boundary.*

Now consider the same analysis on the same dataset with the same pruning method, but with $k = 5000$ and another three stats of interest: hits allowed, strikeouts, and earned run average. The pruning rate over time for this case is plotted (with dashes) in Figure 3 for comparison. We see the pruning rate here picks up more slowly, and converges to a lower percentage. Not only is k bigger in this case, but it also turns out that strikeouts and earned run average are anti-correlated. Together, these factors lead to a more complex decision boundary that requires more samples to acquire, as well as a much lower percentage of perturbations that can be pruned.

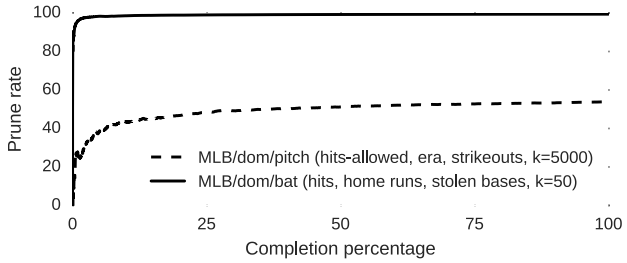


Figure 3: (Adapted from [37].) Pruning rate over the course of serially executing the analysis in Example 5, for different thresholds of k and different stats of interest.

This difference in the pruning rates suggests that the parallel implementations of the two cases above need to be tuned differently to achieve their best performance. When we parallelize the analysis using multiple workers each responsible for a subset of the parameter space, workers can exchange locally cached information to help improve each other's pruning rate. However, some exchanges incur overhead, and their benefits to pruning generally diminish over time, and at different speeds for different workloads. Therefore, workers should exchange information more often early during the execution, and less as time progresses. Furthermore, workers in the first case above can stop exchanging information earlier than in the second case. ■

Thus, when applying parallelization, memoization, and pruning, Perada automatically tunes their performance through run-time observation, learning, and adaptation. To be more specific, Perada divides the parameter space of the perturbation analysis into units, and processes them in parallel on a cluster of workers (using *Spark* [14]). Perada provides both a global key/value cache (using *Redis* [22]) for memoization, and a SQL cache at each worker (using *SQLite* [2]) for pruning, which typically involves more complex queries against the cache. Information exchange through the global key/value cache is immediate, but the contents of all local SQL caches are synchronized only at the end of each execution "epoch," to keep the overhead low. Behind the scene, Perada dynamically and adaptively controls the timing of epochs as well as what combination of optimizations to apply in each epoch. For additional details about Perada, see [37].

From our experience and experiments with real-life applications of perturbation analysis, Perada significantly simplifies development efforts for ad hoc analysis, and delivers vastly better performance than other general solutions based on existing systems. One quibble is that Perada is still not fully declarative; for example, developers have to spell out the pruning logic. As discussed in Section 3.3, we are able to derive pruning logic automatically for some forms of perturbation analysis tasks, but in general, much research is still needed in automatic optimization of declarative analysis.

4 Reflection and Outlook

This paper presents our recent research on query perturbation analysis. We have focused on its applications in fact-checking and lead-finding, and offered an overview of its computational techniques and its connection to traditional query processing. Looking back at our journey, we realize how incredibly lucky we were—we had the chance to collaborate with visionary colleagues in journalism and public policy to work on the important and timely problem of fact-checking, yet, at the same time, the insights and techniques we developed for the problem contribute back to core computer science. Finding new solutions to well-studied, classic database problems, as discussed in Section 3.3, came as a pleasant surprise. Perhaps even more amazing is the experience that we are able to explain the results in our *SIGMOD*, *VLDB*, and *KDD* papers to non-technical friends at parties or strangers on commute, and have them genuinely interested in this research.

In the following, we give a brief chronology of our journey, discuss the broader challenges we faced when building real-world applications, and offer our outlook on the future research in query perturbation analysis.

Project History and Practical Deployments One author of this paper, Yang, was introduced to computational journalism more than ten years ago by James T. Hamilton, a public policy colleague at Duke at the time (now director of the journalism program at Stanford). However, concrete ideas for connecting computational journalism to database research took time to develop, partly because of the overwhelming number of possibilities—computational journalism is so broad that virtually every field of computing can help in various ways. In 2009, Hamilton co-organized a summer workshop on computational journalism at the Center for Advanced Study in the Behavioral Sciences at Stanford, and invited Yang and Sarah Cohen, a seasoned investigative reporter who joined Duke faculty from the Washington Post. This workshop deserves much credit in developing the field; it also helped Yang and Cohen find a focus on data-driven investigative reporting. In parallel, two other authors of this paper, Li and Yu, had begun working on mining interesting facts from data in domains such as sports and weather. A discussion between Yang and Li in the summer of 2010, when both were visiting HP Labs, brought the group together, and identified the connection between fact-checking and lead-finding, as well as the pitfalls of correct but uninteresting or even misleading claims. The group co-authored a vision paper [10] in *CIDR* 2011, in which one could see initial ideas about perturbation analysis starting to take shape.

The project then took off with a diverse research team. Agarwal, an algorithms researcher specializing in computational geometry, added the geometric insights and algorithmic rigor to perturbation analysis. The first PhD student to work on perturbation analysis, You, was co-advised by Yang and Agarwal and focused primarily on the modeling and algorithmic aspects; after graduation, You continued to work on projects related to fact-checking in Cong's group at Google Research. Walenz will be the second PhD student to graduate with a dissertation on perturbation analysis, focusing on system support, practical applications, and most recently, exploring its connection with traditional database query processing together with Roy, a database researcher specializing in theory. Bill Adair, a journalist and fact-checker who founded PolitiFact, joined the Duke faculty in 2013, and has kept the project solidly grounded in its practical fact-checking applications. In retrospect, we recognize that the luck we had was only possible with patience and persistence, as well as a team of collaborators with diverse and complementary backgrounds but the same commitment to practical impact.

Over the years, we have developed a number of applications of perturbation analysis, and worked with real data, real users, as well as journalists and professional fact-checkers. As mentioned earlier, the *iCheck* system, demonstrated at *SIGMOD* 2014 [43], found and checked claims about player performances in professional baseball, publication records of database researchers, and voting records of the US Congress. *FactWatcher*, demonstrated at *VLDB* 2014 [17], found and monitored claims about professional basketball and weather. For the 2016 US elections, we built a website *congress.uclaimicheck.org*. It analyzed the congressional voting records from January 2009 to September 2016, and let visitors compare how legislators voted with party majorities and the President, and more importantly, explore how the comparison stacked up under different contexts using perturbation analysis—over time, among groups of peers, and for "key votes" identified by lobbying organizations. It was demonstrated at the 2016 *Computational+Journalism Symposium* [35] and released to the public in September 2016. Finally, we have been working with Duke Athletics on a system for finding interesting "factlets" about each Duke men's basketball game. It runs after each game, and uses perturbation analysis to produce a variety of factlets about player performances in that game within minutes. As of May 2017, these factlets have become a part of the official Duke men's basketball stats website, available for fans to explore and share on social media.

Challenges in Computational Fact-Checking It has been a humbling experience for us to see the wide gamut of knowledge, skills, and efforts required of journalists in the practical applications that we have worked on. Computational fact-checking poses a broad spectrum of challenges; query perturbation analysis is but one piece of the puzzle. We noticed that when discussing our work with other computer scientists, the first reaction we got

of the form "are you working on X" referred to a wide range of X's. Natural language processing is one of the most often mentioned. Indeed, we need it to map a natural language claim to a query template q. Nonetheless, as seen in Section 2, perturbation analysis can help disambiguate claims; it would be interesting to further explore the idea of using data contents to help understand natural language claims based on them.

Source discovery, data extraction, data integration, and data cleaning remain very labor-intensive tasks. As an example from our experience, even in the case of high-quality congressional voting records, it was a pain collecting and linking other related data, such as key votes published by lobbying organizations. References are often incomplete or ambiguous—especially when many roll calls are associated with the same bill—and worse, occasionally contain typos. Some of them could only be resolved by human experts. While automated data extraction and cleaning techniques have come a long way, attaining the level of accuracy required for fact-checking specific domains without a lot of data—let alone expert-labeled data—remains a challenge in practice. One specific problem that we have been working on recently is how to focus cleaning efforts on subsets of data that matter the most to given fact-checking tasks [31]. Given the complexity of real-life data, we are also interested in developing methods for annotating data that can alert us to potential misuses or misinterpretation.

All of the above challenges—and fact-checking in general—require deep domain knowledge. Consider again Example 1. Without domain knowledge, it is not even clear what "voting with President Obama" means—because US Presidents do not vote—and where we can find Presidential positions on congressional votes. "Voting with democratic majority" is even trickier: the "non-voting" members of the House (e.g., those from Washington D.C. or Guam) do cast votes, but rules about when they get to vote and when their votes get counted kept changing over time depending on which party controlled the House. The few such votes probably never swayed the democratic majority, but just knowing we might be able to ignore this issue also requires domain knowledge. How to capture and utilize such knowledge computationally remains an open challenge.

We have also come to learn how the presentation of claims and fact-checks is critically important to the audience. To illustrate, we do not need to go to examples where perception is influenced by ideologies and personal worldviews; just consider two neutral factlets mined from Duke men's basketball games: 1) "He was one of the only ten players who had at least x points and at least y rebounds in a game in Duke history." 2) "He was one of the only ten players who had at least y rebounds in a game in Duke history." Logically, (2) is stronger. From (2) we can simply tack on the number of rebounds he made in that game, no matter how low it is, and obtain a valid factlet in the form of (1). Furthermore, it is much easier to make a factlet like (1) than (2), because it is less likely for points in higher dimensions to form dominance relations; in other words, (2) would have higher uniqueness than (1). Yet, despite this rational analysis, we found that, anecdotally, the majority of users thought (1) sounded more impressive. This simple example illustrates how no amount of modeling can replace real-world testing, and hints at the challenges of using fact-checking to correct human misperceptions.

Computational fact-checking raises many more challenges. For example, there are also the problems of how to find claims to check in the first place, which we worked on extensively [16, 18], how to deliver fact-checks to the public in a timely manner, which we are starting to research on [3], as well as how to effectively support the collaboration of fact-checkers, journalists, and citizens alike. Last but not least, there is "fake news." Our collaboration with fact-checkers began long before "fake news" rose to prominence in research circles following the 2016 US elections. While we had been busy fighting misleading (but still factually correct) claims, sadly, downright blatant lies were starting to have a field day. Combating such lies requires a different suite of techniques, ranging from source credibility, media forensics, and social network analysis. While it is impossible to give a comprehensive survey on computational fact-checking here, we hope that this discussion, however cursory, will be useful to those considering to work on computational fact-checking and combating misinformation and disinformation in general.

Open Problems in Perturbation Analysis Query perturbation analysis is still a young research direction in databases. We have mentioned a number of open problems throughout this paper. First, there is a wealth

of query templates worthy of consideration. Just pick your favorite lie that levitates numbers—and there are plenty of them out there in media (and research papers!)—and you have a candidate for applying perturbation analysis. Such investigations can be useful and interesting in their own right, and more importantly, they help us discover new ways of assessing claims and countering them, as well as new computational methods, which in turn help us generalize the query perturbation framework and optimization techniques. Second, much work is still needed in supporting fully declarative perturbation analysis and in closing the gap between general and specific solutions. Ideally, we would like to integrate into a database system a comprehensive suite of processing and automatic optimization techniques that provide a level of support for perturbation analysis comparable to that for traditional SQL queries. Directions we are pursuing currently include generic approximation methods, as well as query analysis and rewrite techniques for exploiting more sophisticated pruning opportunities.

This paper has discussed the connection between perturbation analysis and traditional query processing, but there are also connections to many other active areas of database research. For example, a large body of work on *uncertain data management* [11, 5] considers the effect of data perturbations on query results. Our study of query parameter perturbations can be seen as a conceptual counterpoint—while uncertain databases consider one query over many database instances (possible worlds), we are also interested in many queries (perturbed versions of each other) over one database instance. There has also been much work on *lineage* and *provenance* (surveyed in [8]), and more recently, *causality* [23, 24, 25, 26]. This line of work also studies how data contribute to query results; depending on how contribution is defined, some problems may be analogous or related to the analysis of query parameter perturbations. Recent work on *query by output* [34] and *view synthesis* [28] can be seen as types of perturbation analysis, where we search for queries with desired properties. Also similar in spirit are [33, 19, 38], though their search spaces and goals are different. Finally, *differential privacy* [12] has an element of understanding sensitivity as well—how an individual data perturbation can affect query results in the worst case. It would be very interesting to further explore potentially deep connections among various types of perturbations of data, parameters, and query forms.

Conclusion The biggest takeaway point about query perturbation analysis can be perhaps summarized by a phrase from the title of You's dissertation—it is a change of perspective "from answering questions to questioning answers and raising good questions," and this change not only introduces new database research challenges but also brings new insights into well-studied problems. Perturbation analysis was initially motivated by fact-checking, and we continue to collaborate closely with fact-checkers and journalists to push the boundaries of computational fact-checking. Speaking from our experience, working as an interdisciplinary team to tackle problems of societal importance can be both practically fulfilling and intellectually stimulating.

Acknowledgments *This work was partially supported by NSF grants IIS-1408846, IIS-1408928, IIS-1565699, IIS-1719054, grants from the Knight Foundation, and a Google Faculty Award. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the funding agencies.*

References

[1] PostgreSQL. `http://www.postgresql.org/`.

[2] SQLite. `http://sqlite.org/`.

[3] B. Adair, C. Li, J. Yang, and C. Yu. Progress toward "the holy grail": The continued quest to automate fact-checking. In *Proc. 2017 Computation+Journalism Symposium*, Evanston, Illinois, USA, Oct. 2017. Informal publication.

[4] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan. Geometric approximation via coresets. In J. E. Goodman, J. Pach, and E. Welzl, editors, *Combinatorial and Computational Geometry*, pages 1–30. Cambridge University Press, New York, 2005.

[5] C. C. Aggarwal, editor. *Managing and Mining Uncertain Data*. Springer, 2009.

[6] R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *Proc. 1994 Intl. Conf. on Very Large Data Bases*, pages 487–499, Santiago de Chile, Chile, Sept. 1994.

[7] S. Börzsönyi, D. Kossmann, and K. Stocker. The skyline operator. In *Proc. 2001 Intl. Conf. on Data Engineering*, pages 421–430, Heidelberg, Germany, Apr. 2001.

[8] J. Cheney, L. Chiticariu, and W. C. Tan. Provenance in databases: Why, how, and where. *Foundations and Trends in Databases*, 1(4):379–474, 2009.

[9] S. Cohen, J. T. Hamilton, and F. Turner. Computational journalism. *Communications of the ACM*, 54(10):66–71, 2011.

[10] S. Cohen, C. Li, J. Yang, and C. Yu. Computational journalism: A call to arms to database researchers. In *Proc. 2011 Conf. on Innovative Data Systems Research*, Asilomar, California, USA, Jan. 2011.

[11] N. N. Dalvi, C. Ré, and D. Suciu. Probabilistic databases: Diamonds in the dirt. *Communications of the ACM*, 52(7):86–94, 2009.

[12] C. Dwork. A firm foundation for private data analysis. *Communications of the ACM*, 54(1):86–95, 2011.

[13] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J. D. Ullman. Computing iceberg queries efficiently. In *Proc. 1998 Intl. Conf. on Very Large Data Bases*, pages 299–310, New York City, New York, USA, Aug. 1998.

[14] T. A. S. Foundation. Apache Spark: Lightning-fast cluster computing. `http://spark.apache.org/`.

[15] P. González, A. C. no, N. V. Chawla, and J. J. D. Coz. A review on quantification learning. *ACM Computing Surveys*, 50(5):74:1–74:40, Sept. 2017.

[16] N. Hassan, F. Arslan, C. Li, and M. Tremayne. Toward automated fact-checking: Detecting check-worthy factual claims by claimbuster. In *Proc. 2017 ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, Halifax, Nova Scotia, Canada, Aug. 2017.

[17] N. Hassan, A. Sultana, Y. Wu, G. Zhang, C. Li, J. Yang, and C. Yu. Data in, fact out: Automated monitoring of facts by FactWatcher. *Proc. the VLDB Endowment*, 7(13), 2014.

[18] N. Hassan, G. Zhang, F. Arslan, J. Caraballo, D. Jimenez, S. Gawsane, S. Hasan, M. Joseph, A. Kulkarni, A. K. Nayak, V. Sable, C. Li, and M. Tremayne. ClaimBuster: The first-ever end-to-end fact-checking system. *Proc. the VLDB Endowment*, 10(12), Aug. 2017.

[19] Z. He and E. Lo. Answering why-not questions on top-k queries. In *Proc. 2012 Intl. Conf. on Data Engineering*, pages 750–761, Washington DC, USA, Apr. 2012.

[20] D. Huff. *How to Lie with Statistics*. W. W. Norton & Company, reissue edition, 1993.

[21] L. Khachiyan. *Fourier–Motzkin elimination method*, pages 1074–1077. Encyclopedia of Optimization, Springer US, Boston, MA, 2009.

[22] R. Labs. Redis. `http://redis.io/`.

[23] A. Meliou, W. Gatterbauer, J. Y. Halpern, C. Koch, K. F. Moore, and D. Suciu. Causality in databases. *IEEE Data Engineering Bulletin*, 33(3):59–67, 2010.

[24] A. Meliou, W. Gatterbauer, K. F. Moore, and D. Suciu. The complexity of causality and responsibility for query answers and non-answers. *Proc. the VLDB Endowment*, 4(1):34–45, 2010.

[25] A. Meliou, W. Gatterbauer, K. F. Moore, and D. Suciu. WHY SO? or WHY NO? functional causality for explaining query answers. In *Proc. 2010 Intl. VLDB Workshop on Management of Uncertain Data*, pages 3–17, Singapore, Sept. 2010.

[26] A. Meliou, W. Gatterbauer, and D. Suciu. Bring provenance to its full potential using causal reasoning. In *Proc. 2011 USENIX Workshop on the Theory and Practice of Provenance*, Heraklion, Crete, Greece, June 2011.

[27] D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive skyline computation in database systems. *ACM Transactions on Database Systems*, 30(1):41–82, 2005.

[28] A. D. Sarma, A. G. Parameswaran, H. Garcia-Molina, and J. Widom. Synthesizing view definitions from data. In *Proc. 2010 Intl. Conf. on Database Theory*, pages 89–103, Lausanne, Switzerland, Mar. 2010.

[29] T. K. Sellis. Multiple-query optimization. *ACM Transactions on Database Systems*, 13(1):23–52, 1988.

[30] B. Settles. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2012.

[31] S. Sintos, P. K. Agarwal, and J. Yang. Data cleaning and fact checking: Minimizing uncertainty versus maximizing surprise. Technical report, Duke University, 2018. `https://users.cs.duke.edu/~ssintos/DataCleanFactCheck.pdf`.

[32] S. T. Tokdar and R. E. Kass. Importance sampling: A review. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2:54–60, 2010.

[33] Q. T. Tran and C.-Y. Chan. How to ConQueR why-not questions. In *Proc. 2010 ACM SIGMOD Intl. Conf. on Management of Data*, pages 15–26, Indianapolis, Indiana, USA, June 2010.

[34] Q. T. Tran, C.-Y. Chan, and S. Parthasarathy. Query by output. In *Proc. 2009 ACM SIGMOD Intl. Conf. on Management of Data*, pages 535–548, Providence, Rhode Island, USA, June 2009.

[35] B. Walenz, J. Gao, E. Sonmez, Y. Tian, Y. Wen, C. Xu, B. Adair, and J. Yang. Fact checking congressional voting claims. In *Proc. 2016 Computation+Journalism Symposium*, Stanford, California, USA, Sept. 2016. Informal publication.

[36] B. Walenz, S. Roy, and J. Yang. Optimizing iceberg queries with complex joins. In *Proc. 2017 ACM SIGMOD Intl. Conf. on Management of Data*, pages 1243–1258, Chicago, Illinois, USA, May 2017.

[37] B. Walenz and J. Yang. Perturbation analysis of database queries. *Proc. the VLDB Endowment*, 9(14):1635–1646, Sept. 2016.

[38] E. Wu and S. Madden. Scorpion: Explaining away outliers in aggregate queries. *Proc. the VLDB Endowment*, 6(8):553–564, June 2013.

[39] Y. Wu, P. K. Agarwal, C. Li, J. Yang, and C. Yu. On "one of the few" objects. In *Proc. 2012 ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, pages 1487–1495, Beijing, China, Aug. 2012.

[40] Y. Wu, P. K. Agarwal, C. Li, J. Yang, and C. Yu. Toward computational fact-checking. *Proc. the VLDB Endowment*, 7(7):589–600, 2014.

[41] Y. Wu, P. K. Agarwal, C. Li, J. Yang, and C. Yu. Computational fact checking through query perturbations. *ACM Transactions on Database Systems*, 42(1):4:1–4:41, Mar. 2017.

[42] Y. Wu, J. Gao, P. K. Agarwal, and J. Yang. Finding diverse, high-value representatives on a surface of answers. *Proc. the VLDB Endowment*, 10(7):793–804, Mar. 2017.

[43] Y. Wu, B. Walenz, P. Li, A. Shim, E. Sonmez, P. K. Agarwal, C. Li, J. Yang, and C. Yu. iCheck: Computationally combating "lies, d—ned lies, and statistics". In *Proc. 2014 ACM SIGMOD Intl. Conf. on Management of Data*, Snowbird, Utah, USA, June 2014.

Explanation Tables

Kareem El Gebaly
University of Waterloo
kelgebaly@uwaterloo.ca

Guoyao Feng
CMU
gfeng@andrew.cmu.edu

Lukasz Golab
University of Waterloo
lgolab@uwaterloo.ca

Flip Korn
Google Research
flip@google.com

Divesh Srivastava
AT&T Labs - Research
divesh@research.att.com

Abstract

We present a robust solution to the following problem: given a table with multiple categorical dimension attributes and one binary outcome attribute, construct a summary that offers an interpretable explanation of the factors affecting the outcome attribute in terms of the dimension attribute value combinations. We refer to such a summary as an explanation table*, which is a disjunction of overlapping patterns over the dimension attributes, where each pattern specifies a conjunction of attribute=value conditions. The* Flashlight *algorithm that we describe is based on sampling and includes optimizations related to computing the information content of a summary from a sample of the data. Using real data sets, we demonstrate the advantages of explanation tables compared to related approaches that can be adapted to solve our problem, and we show significant performance benefits of our approach.*

1 Introduction

Before making complex data-driven decisions, users often wish to obtain a summary of the available data sets to understand their properties; this is particularly important for auditing prior knowledge and/or policy, for example to ensure fairness. Motivated by this need, we present a robust solution to the following problem: given a table T with multiple categorical *dimension attributes* and one binary *outcome attribute* b, how can we construct a summary that offers an interpretable explanation of the factors affecting the binary outcome attribute in terms of the dimension attribute value combinations?

Consider the following example, drawn from [6]. An athlete may be interested in deciding when to exercise and what to eat before exercising to meet her target goals. To make these decisions in a data-driven fashion, she monitors her activities and maintains an exercise log. Table 1 shows her exercise log, with each row containing an ID, the time and day of the week of the exercise, the meal eaten before the exercise, and a binary attribute indicating the outcome of the exercise, i.e., whether a target goal was met. The three dimension attributes in this example are `day`, `time` and `meal`.

Table 2 illustrates a corresponding summary. It contains a list of *patterns* that specify subsets of tuples, where a pattern is a tuple from the data cube over the dimension attributes of T. Patterns consist of attribute values or the wildcard symbol "*" denoting all possible values. Each pattern is associated with the number of

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Table 1: An Exercise relation

id	day	time	meal	goal met?
1	Fri	Dawn	Banana	1
2	Fri	Night	Green salad	1
3	Sun	Dusk	Oatmeal	1
4	Sun	Morning	Banana	1
5	Mon	Afternoon	Oatmeal	1
6	Mon	Midday	Banana	1
7	Tue	Morning	Green salad	0
8	Wed	Night	Burgers	0
9	Thu	Dawn	Oatmeal	1
10	Sat	Afternoon	Nuts	0
11	Sat	Dawn	Banana	0
12	Sat	Dawn	Oatmeal	0
13	Sat	Dusk	Rice	0
14	Sat	Midday	Toast	0

Table 2: An explanation table over the Exercise relation

day	time	meal	goal met?	count
*	*	*	.5	14
Sat	*	*	0	5
*	*	Banana	.75	4
*	*	Oatmeal	.75	4

tuples it matches (count) and a fraction indicating how many of these tuples have the binary outcome attribute equal to one.

The first pattern in Table 2 matches all the tuples in Table 1 and indicates that, on average, half the workouts were successful. The second pattern reveals that all Saturday workouts were unsuccessful, which provides the most information about the outcome attribute beyond what we know so far from the first pattern. Intuitively, this is because the average value of the goal met attribute on Saturdays is significantly different than the average value of this attribute in the entire dataset, and there are sufficiently many Saturday tuples in the dataset. The last two patterns state that eating bananas or oatmeal led to successful workouts 75 percent of the time. Note that patterns may overlap. For example, the patterns (*,Morning,*) and (*,*,Banana) overlap, suggesting that working out in the morning after eating a banana could influence the distribution of the outcome attribute differently from working out in the morning or working out after eating a banana.

A natural language for expressing summaries over multi-dimensional data is in terms of a disjunction of patterns over the dimension attributes, where each pattern specifies a conjunction of "attribute=value" conditions. This form of expression has appeared extensively in the data mining and OLAP literature, from datacubes to association rules, and has been employed more specifically in settings where an "explanation" of some property or outcome attribute in a data set in terms of the dimensions is needed; see [3, 4, 8, 12, 14] for examples. However, since these existing explanatory summaries are not always grounded in information theory, we argue that they are not fully optimized for *interpretability* in the following sense. We define the *accuracy* of an explanation E in terms of its information loss, quantified by the relative entropy with respect to some table T that it summarizes, where less information loss implies more accuracy. Since there is an inherent trade-off between accuracy and

explanation size, interpretability is a relative notion. Therefore, given two explanations, E and E', where E is at least as accurate as E' and E has fewer patterns than E' (i.e., $|E| < |E'|$), by Occam's Razor we should prefer E since it is the more *concise* one.[1]

We take a summary with fewer (disjunctive) patterns as being more concise, rather than the total number of "attribute=value" conditions, because it provides explanations that are data-driven rather than schema-driven. For example, when many tuples in T that are concentrated in a very narrow subspace have uniform labels, it is desirable to include a pattern containing them in a summary, but counting the number "attribute=value" conditions would penalize heavily for this pattern. Since negations are not used in patterns, intuitively the way that we are able to achieve conciseness is by allowing the patterns to *overlap* on the data set records that they match. In contrast, many existing methods such as decision trees use disjoint patterns, which can result in significantly more complex explanations (e.g., due to negations). Among the existing approaches that allow overlap, the semantics of the overlap are not rich enough to provide conciseness, as we demonstrate experimentally in this paper.

2 Explanation Tables

Our proposed summary is called *Explanation Tables*, an example of which was provided in Table 2 for the data set in Table 1. Next, we explain our algorithm for computing explanation tables; see [6, 5, 7] for further details and extensions to numeric outcome attributes, distributed computation and in-browser computation using Javascript.

2.1 Computing Explanation Tables

The main idea is to maintain *maximum-entropy estimates* $\hat{b}$ for the values of the outcome attribute b and refine them as new patterns are added to the explanation table. Let $t[b]$ and $t[\hat{b}]$ be the actual and estimated values of b, respectively, for a tuple $t \in T$. We use *Kullback-Leibler Divergence* (KL-Divergence) to quantify the difference between the actual and estimated distributions, which is defined as follows for binary attributes: $KL(b||\hat{b}) = \sum_{t \in T | t[b]=1} \log\left(\frac{1}{t[\hat{b}]}\right) + \sum_{t \in T | t[b]=0} \log\left(\frac{1}{1-t[\hat{b}]}\right)$. In each iteration of the algorithm, we greedily add a pattern to the explanation table that provides the most information gain, i.e., leads to the greatest reduction in KL-Divergence. We stop after k iterations, where k is a user-supplied parameter, yielding an explanation table with k patterns.

Recall Table 2. Based on the first pattern (the all-stars pattern), the maximum-entropy estimate of the outcome attribute is to set each value (of each of the 14 rows in Table 1) to 0.5. Of course, the actual values of the `goal met` attribute are binary, but we allow the estimates to be real numbers between zero and one. This maximum-entropy estimate only uses the information implied by the first pattern of the explanation table, which is that `AVG(goal met)=0.5`, without making any other assumptions.

As it turns out, the pattern (Sat,*,*) is then added to the explanation table because it provides the greatest information gain. As a result, we update the estimates $t[\hat{b}]$ as follows. Since the second pattern implies that all Saturday tuples have `goal met=0`, we set the estimates for rows 10 through 14 in Table 1 to zero. Next, for consistency with the first pattern, which requires the average value of the outcome attribute over the whole dataset to be 0.5, we must set the estimates for all non-Saturday rows to $\frac{7}{9}$ each. This gives us a maximum-entropy estimate that only considers the information contained in the first two patterns of the explanation table.

Since the patterns of an explanation table may overlap, it is not obvious how to compute a maximum-entropy estimate in the general case. We use an approximation technique called *iterative scaling* [1, 2] to compute updated estimates whenever a new pattern is added to the explanation table.

[1] Usually the goal is not to (losslessly) capture every detail in the data set but represent it at some granularity that achieves a balance between filtering the noise while preserving the signal.

Algorithm 1 Greedy algorithm template for constructing explanation tables

Require: T, k

1: $E \leftarrow (*, \ldots, *)$
2: **for** each $t \in T$ **do**
3: $\quad t[\hat{b}] \leftarrow$ fraction of rows in T with $t[b] = 1$
4: **end for**
5: **for** $i = 1$ to $k - 1$ **do**
6: $\quad P \leftarrow$ set of candidate patterns
7: $\quad p_{opt} \leftarrow \arg\max_{p \in P}$ information gain(p)
8: $\quad E \leftarrow E \cup p_{opt}$
9: $\quad$ update $t[\hat{b}]$ for each $t \in T$ using iterative scaling
10: **end for**

Algorithm 1 shows a greedy algorithm template for constructing an explanation table E with k patterns from table T. We iterate k times, once to obtain each pattern, with the all-stars pattern added first. In each subsequent iteration, we identify a set P of candidate patterns, we then add to E the pattern from P with the highest information gain, and we finally update the maximum-entropy estimates $t[\hat{b}]$ via iterative scaling.

2.2 Sampling to Improve Efficiency

A naive instantiation of the template of Algorithm 1 considers all possible patterns in line 6 during each iteration. This may not be feasible for large datasets, both in the number of rows and the number of columns. To improve efficiency, we *prune* the candidate space by drawing a random sample s from T in line 6 (a different sample in each iteration of the loop) and setting P to be only the patterns from the data cube of the sample. The intuition is that patterns with frequently occurring combinations of dimension attribute values are likely to be sampled and also to have high information gain. In line 7, we compute the gain of only the patterns in P, as described below.

We illustrate our solution using Table 1. Suppose that the pattern (*,*,*) has been added to the explanation table, meaning that $t[\hat{b}] = 0.5$ for each tuple in T. Suppose that we draw a sample s that consists of tuples 4, 9 and 11 from Table 1.

The first step is to compute the *Lowest Common Ancestor* (LCA) of each pair of tuples having one tuple from s and one from T. To compute the LCA, non-matching (categorical) attribute values are replaced with stars. For example, the LCA of (Sun,Morning,Banana) and (Fri,Dawn,Banana) is (*,*,Banana). Table 3 shows all the LCAs in our example, including the same LCAs computed from different pairs of tuples. Each row of Table 3 corresponds to one of the 14 tuples from T and each column corresponds to one of the three tuples from the sample s.

In the second step, we do a group-by over the LCAs, and, for each distinct pattern, we compute the three aggregates shown in Table 4: count, sum($t[b]$) and sum($t[\hat{b}]$). Ignore "Ancestors" for now; we will explain them in the next step. To understand these three aggregates, consider (*,*,Banana). It was generated as an LCA five times, as shown in Table 3. Thus, its count is five. To understand sum($t[b]$), note that the five occurrences of (*,*,Banana) in the LCA set are derived from (Fri,Dawn,Banana) which has $t[b] = 1$, (Sun,Morning,Banana) which has $t[b] = 1$, (Mon,Midday,Banana) twice which has $t[b] = 1$, and finally (Sat,Dawn,Banana) which has $t[b] = 0$. The sum of these $t[b]$'s is four, including counting (Mon,Midday,Banana) twice. Similarly, the sum of the $t[\hat{b}]$'s is 2.5, including counting (Mon,Midday,Banana) twice; recall that at this point every tuple in T has $t[\hat{b}] = 0.5$.

In the third step, we generate the *ancestors* of each distinct LCA pattern p, defined as p itself plus all other patterns that are the same as p except one or more of the attribute values in p have been replaced by wildcards. These are shown in the rightmost column of Table 4. Each ancestor of p is associated with the same count,

Table 3: Computing LCAs (first step of the algorithm)

T \s	(Sun,Morning,Banana)	(Thu,Dawn,Oatmeal)	(Sat,Dawn,Banana)
(Fri,Dawn,Banana)	(*,*,Banana)	(*,Dawn,*)	(*,Dawn,Banana)
(Fri,Night,Green salad)	(*,*,*)	(*,*,*)	(*,*,*)
(Sun,Dusk,Oatmeal)	(Sun,*,*)	(*,*,Oatmeal)	(*,*,*)
(Sun,Morning,Banana)	(Sun,Morning,Banana)	(*,*,*)	(*,*,Banana)
(Mon,Afternoon,Oatmeal)	(*,*,*)	(*,*,Oatmeal)	(*,*,*)
(Mon,Midday,Banana)	(*,*,Banana)	(*,*,*)	(*,*,Banana)
(Tue,Morning,Green salad)	(*,Morning,*)	(*,*,*)	(*,*,*)
(Wed,Night,Burgers)	(*,*,*)	(*,*,*)	(*,*,*)
(Thu,Dawn,Oatmeal)	(*,*,*)	(Thu,Dawn,Oatmeal)	(*,Dawn,*)
(Sat,Afternoon,Nuts)	(*,*,*)	(*,*,*)	(Sat,*,*)
(Sat,Dawn,Banana)	(*,*,Banana)	(*,Dawn,*)	(Sat,Dawn,Banana)
(Sat,Dawn,Oatmeal)	(*,*,*)	(*,Dawn,Oatmeal)	(Sat,Dawn,*)
(Sat,Dusk,Rice)	(*,*,*)	(*,*,*)	(Sat,*,*)
(Sat,Midday,Toast)	(*,*,*)	(*,*,*)	(Sat,*,*)

sum($t[b]$) and sum($t[\hat{b}]$) values as p. We then take all the ancestors and compute another group-by on each unique pattern to get new count, sum($t[b]$) and sum($t[\hat{b}]$) values. These are shown in the four leftmost columns of Table 5. For example, (*,*,Banana) is an ancestor of itself (with count 5), of (*,Dawn,Banana) (with count 1), of (Sat,Dawn,Banana) (with count 1) and of (Sun,Morning,Banana) (with count 1). Thus, its count in Table 5 is eight, and sum($t[b]$) and sum($t[\hat{b}]$) are computed similarly.

Note that at this point, after generating the ancestors of all the LCAs, the resulting set of 20 patterns is exactly the data cube of s. This is our candidate set P referenced in line 6 of the algorithm.

In the fourth step, we "correct" the counts and sums so that they correspond to the true count of tuples matching each pattern and the true sums of the matching $t[b]$ and $t[\hat{b}]$ values. To do this, we consider the number of tuples in the sample that match each pattern. For example (*,*,Banana) matches two tuples in s, namely tuples 4 and 11, so we divide the aggregates we have computed for (*,*,Banana) by two. This gives the corrected $count = 4$, $sum(t[b]) = 3$ and $sum(t[\hat{b}]) = 2$. The rightmost four columns of Table 5 show the frequency in sample of each candidate pattern p (i.e., how many tuples in s it matches) and the corrected aggregates.

Finally, we select the pattern with the highest information gain based on the computed aggregates. We approximate the information gain of a pattern p as $count \times (sum(t[b]) \log \frac{sum(t[b])}{sum(t[\hat{b}])} + (1 - sum(t[b])) \log \frac{1-sum(t[b])}{1-sum(t[\hat{b}])})$. In our example, the pattern (Sat,*,*) has the highest information gain of $5 \times (0 + \log(\frac{1}{0.5})) = 5 \log 2$, so it is added to the explanation table.

The computational complexity of our approach is $O(|s| \times |T|)$ to compute the LCAs and their ancestors, plus $O(|datacube(s)|)$ to generate ancestors, plus $O(|datacube(s)| \times |s|)$ to join the patterns with the sample and compute the corrected aggregates. Note that $|datacube(s)|$, the size of the data cube of the sample s, is likely to be much smaller than $|datacube(T)|$. In fact, based on the results we reported in [6], our approach (called the "Flashlight Strategy" in that work) can be orders of magnitude faster than computing the information gain of all possible patterns in $datacube(T)$, with only slightly worse information gain.

An important aspect of our sampling-based approach is that sampling is used only to restrict the space of candidate patterns, not to compute their information gain. In other words, the $sum(t[b])$ and $sum(t[\hat{b}])$ aggregates are computed over the underlying dataset, not just the sample. In [6], we also investigated an even faster approach, termed "Laserlight", in which these aggregates are computed over the sample. This is faster

Table 4: Computing aggregates over the LCAs from Table 3 (second step of the algorithm) and generating ancestors (third step of the algorithm)

Pattern	count	sum($t[b]$)	sum($t[\hat{b}]$)	Ancestors
(*,*,*)	21	9	10.5	(*,*,*)
(*,*,Banana)	5	4	2.5	(*,*,Banana), (*,*,*)
(*,Dawn,*)	3	2	1.5	(*,Dawn,*), (*,*,*)
(Sat,*,*)	3	0	1.5	(Sat,*,*), (*,*,*)
(*,*,Oatmeal)	2	2	1	(*,*,Oatmeal), (*,*,*)
(*,Dawn,Banana)	1	0	.5	(*,Dawn,Banana), (*,Dawn,*), (*,*,Banana), (*,*,*)
(*,Dawn,Oatmeal)	1	0	.5	(*,Dawn,Oatmeal), (*,Dawn,*), (*,*,Oatmeal), (*,*,*)
(*,Morning,*)	1	0	.5	(*,Morning,*), (*,*,*)
(Sat,Dawn,*)	1	0	.5	(Sat,Dawn,*), (Sat,*,*),(*,Dawn,*),(*,*,*)
(Sat,Dawn,Banana)	1	0	.5	(Sat,Dawn,Banana), ..., (Sat,*,*), (*,Dawn,*),(*,*,*)
(Sun,*,*)	1	1	.5	(Sun,*,*), (*,*,*)
(Sun,Morning,Banana)	1	1	.5	(Sun,Morning,Banana), ..., (*,*,Banana), (*,*,*)
(Thu,Dawn,Oatmeal)	1	1	.5	(Thu,Dawn,Oatmeal), ..., (*,*,Oatmeal),(*,*,*)

because it does not require a join with the underlying table T to compute the LCAs; instead it simply computes a data cube over the sample s. However, we observed in [6] that the improvement in efficiency of "Laserlight" comes at a steep cost of information gain, making the strategy we described here ("Flashlight") a more effective choice for constructing informative explanation tables.

3 Experimental Results

In this section, we provide a brief experimental evaluation of the information content of explanation tables compared to related techniques. The experiments were performed on a 3.3 GHz AMD machine with 16 GB of Ram and 6 MB of cache, running Ubuntu Linux. We use the following datasets.

- `Upgrade`: 5795 United Airline ticket records obtained from `https://www.diditclear.com/`. Each record contains seven attributes about the tickets (origin airport, destination airport, date, booking class, etc.) and a binary attribute indicating whether the passenger was upgraded to business class. This data set has over 346,000 possible patterns.

- `Adult`: U.S. Census data containing demographic attributes such as occupation, education and gender, and a binary outcome attribute denoting whether the given person's income exceeded \$50K. This data set was downloaded from the UCI archive at `archive.ics.uci.edu/ml/datasets/Adult/` and contains 32561 tuples, 8 attributes and $436,414$ possible patterns.

We measured the information content of explanation tables and other summaries in terms of the *average information gain*. This is the improvement in KL-Divergence compared to having only the all-wildcards pattern and knowing only the fraction of tuples in the whole data set having outcome 1.

To compute explanation tables, we used the sampling-based approach from [6] which we described earlier, with a sample size of 16, which we observed to offer a good trade-off between running time and information gain on the tested datasets (we observed diminishing returns in terms of information content for larger sample sizes). The reported information gains are averages of five runs with different samples.

We compare explanation tables with four related approaches. The first three were compared in [6], whereas a comparison with the last one is a novel contribution of this paper.

Table 5: Computing aggregates over the LCAs and their ancestors (fourth step of the algorithm)

Pattern	count	sum($t[b]$)	sum($t[\hat{b}]$)	Frequency in sample	$count$	sum($t[b]$)	sum($t[\hat{b}]$)
(*,*,*)	42	21	21	3	14	7	7
(*,*,Banana)	8	6	4	2	4	3	2
(*,*,Oatmeal)	4	3	2	1	4	3	2
(*,Dawn,*)	8	4	4	2	4	2	2
(*,Morning,*)	2	1	1	1	2	1	1
(Sat,*,*)	5	0	2.5	1	5	0	2.5
(Sun,*,*)	2	2	1	1	2	2	1
(Thu,*,*)	1	1	.5	1	1	1	.5
(*,Dawn,Banana)	2	1	1	1	2	1	1
(*,Dawn,Oatmeal)	2	1	1	1	2	1	1
(*,Morning,Banana)	1	1	.5	1	1	1	.5
(Sat,*,Banana)	1	0	.5	1	1	0	.5
(Sun,*,Banana)	1	1	.5	1	1	1	.5
(Thu,*,Oatmeal)	1	1	.5	1	1	1	.5
(Sat,Dawn,*)	2	0	1	1	2	0	1
(Sun,Morning,*)	1	1	.5	1	1	1	.5
(Thu,Dawn,*)	1	1	.5	1	1	1	.5
(Sun,Morning,Banana)	1	1	.5	1	1	1	.5
(Sat,Dawn,Banana)	1	0	.5	1	1	0	.5
(Thu,Dawn,Oatmeal)	1	1	.5	1	1	1	.5

1. Decision trees, as an example of a human-interpretable classifier. Each root-to-leaf path in a decision tree can be thought of as a pattern (of values of the dimension attributes), with the leaf node providing a prediction for the value of the outcome attribute. We computed the information gain of decision trees by comparing the predicted values of the outcome attribute with the true values.

   We used the Weka J48 implementation [10] of the C4.5 algorithm, which provides a *confidence* parameter to control pruning and the number of tuples a leaf node can cover. We considered both multi-way split decision trees in which a node can have many children and binary split decision trees in which a node has exactly two children. Since there is no direct way to enforce decision tree size in Weka, we performed a grid search over the parameter space to obtain trees with specific numbers of root-to-leaf paths. To do this, in our experiments, we varied confidence in the range $[0.01 - 0.59]$ and the minimum number of tuples per leaf in the range $[2 - 20]$.

2. The SURPRISE operator for data cube exploration [13], which identifies regions of a datacube whose distribution of the outcome variable is different than the distribution in the regions already seen by the user. As in explanation tables, these regions are represented by patterns of values of the dimension attributes. However, while SURPRISE also uses information gain to find informative patterns, for efficiency reasons it only considers non-overlapping patterns or patterns that are fully contained in each other (corresponding to rolling-up or drilling-down in the datacube). We implemented this technique in C++ based on the algorithm given in [13]. For a fair comparison with explanation tables, we start with the knowledge of the average value of the outcome attribute in the dataset, and we request the k most informative patterns.

   The SURPRISE operator includes two parameters: the desired number of patterns and an accuracy threshold ϵ. A lower ϵ indicates a finer algorithm resolution but slower runtime. To fairly represent SURPRISE, we have chosen two values for ϵ for each data set: SURPRISE_cmp, which uses ϵ that gives comparable

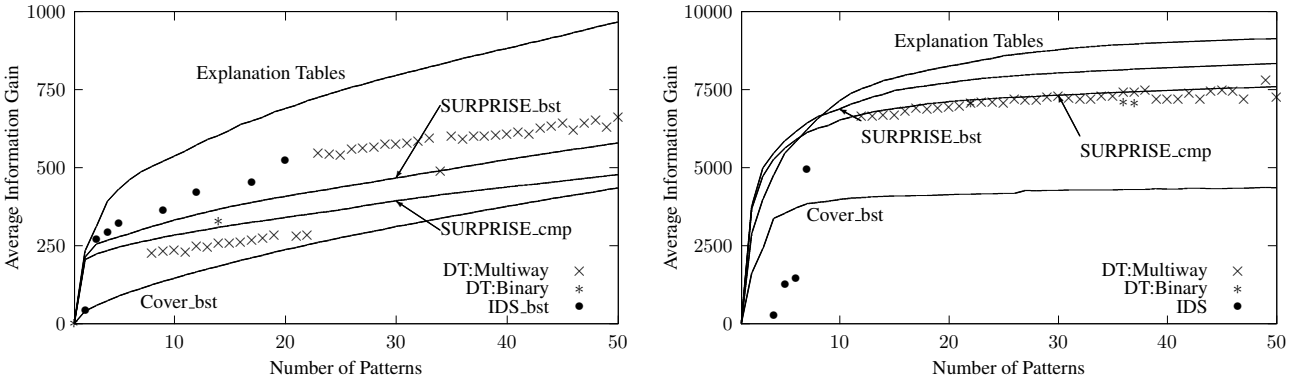


Figure 1: Average information gain of various methods using `Upgrade` (left) and `Adult` (right)

runtime to *Flashlight*, and SURPRISE_bst, which uses ϵ that yields the best possible information gain without running out of memory.

3. Data summarization techniques from [8, 9] which find the fewest patterns that cover most of the tuples with outcome 1. These methods require a confidence threshold that denotes the minimum fraction of 1-outcomes per pattern. We experiment with a variety of confidence thresholds, including $\{0.4, 0.5, \ldots, 0.9, 0.95, 0.99\}$, and report the runtime of the one with the best information gain, denoted Cover_bst. After generating the summaries, we apply the maximum entropy principle to derive an estimated distribution of the outcome attribute and we compute information gain based on this.

4. Interpretable Decision Sets (IDS) [11], as another recent example of a human-interpretable classifier. An IDS is a set of independent if-then classification rules. We obtained the source code from the authors' Github repository[2]. The IDS generation algorithm includes a support threshold ϵ, which determines the lowest possible support of a given classification rule. We started with $\epsilon = 1$ and considered lower values until the runtime became too long. We varied ϵ in decrements of 0.02 until the runtime exceeded one hour for the `Upgrade` dataset and three hours for the larger `Adult` dataset. As with decision trees, we computed the information gain of IDS by comparing the predicted values of the outcome attribute with the true values.

Figure 1 plots the average information gain using `Upgrade` (left) and `Adult` (right) as a function of the number of patterns (or, in the case of decision trees, the number of leaves). Decision trees are represented as individual points on the graphs, corresponding to the numbers of leaves we were able to obtain using the grid search explained above. Similarly, IDS results are represented as individual points that indicate the best possible information gain we could obtain using the above grid search. On the other hand, SURPRISE and Cover allow us to control the number of patterns, so these are represented as lines. We conclude that explanation tables outperform all other approaches in terms of information gain, except for small numbers of patterns (below 8) on the `Adult` data set, in which case SURPRISE has a slightly higher information gain. This is likely due to the fact that explanation tables explicitly optimize for information content. On the other hand, decision trees and SURPRISE are restricted to non-overlapping patterns; Cover optimizes for the number of patterns that cover all the positive examples without considering information gain; and IDS is a classifier that optimizes jointly for predictive power, conciseness and interpretability.

In terms of running time, it took 47 seconds on `Upgrade` and 252 seconds on `Adult` to generate explanation tables with 50 patterns. SURPRISE_cmp took roughly the same time (by design), whereas SURPRISE_bst

[2]https://github.com/lvhimabindu/interpretable_decision_sets

was 2-4 times slower and Cover_bst was 5-8 times slower. On the other hand, decision trees and IDS took much longer (several hours) since they had to be executed multiple times with different parameter values in order to obtain a desired number of patterns.

4 Discussion

We emphasize that our problem is different from prediction since explanation conciseness is also critical. Hence, instead of focusing on prediction accuracy we focus on interpretability, which is a relative notion that captures an inherent trade-off between accuracy and conciseness: given two explanations, E and E', if E is at least as accurate as E' and E is more concise than E', we say that E is more interpretable than E'. While there exist some classifiers that could be considered human-interpretable as a secondary goal (e.g., decision trees and partial monotonic functions), in practice they are used in ensembles which hinders their interpretability. Therefore, explanation tables may have a potentially interesting role to play when the outcome attribute corresponds to the predictions from an ensemble method, or from an even more opaque model such as variants of neural networks, since they may enable auditing or debugging the model. This application would be an interesting direction to explore for future work. Future work could also include a user study to evaluate the explanatory power of explanation tables for various tasks (e.g., why was this example classified as such?) as well as the extension to (unordered) categorical outcome attributes.

References

[1] A. Berger: The improved iterative scaling algorithm: A gentle introduction, 1997.

[2] J. N. Darroch and D. Ratcliff: Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 1470-1480, 1972.

[3] M. Das, S. Amer-Yahia, G. Das and C.Yu: MRI: Meaningful Interpretations of Collaborative Ratings, 1063–1074, 2011.

[4] D. Fabbri and K. LeFevre: Explanation-Based Auditing. *P*VLDB 5(1): 1–12 (2012)

[5] G. Feng, L. Golab, D. Srivastava: Scalable Informative Rule Mining. ICDE 2017: 437-448

[6] K. El Gebaly, P. Agrawal, L. Golab, F. Korn, D. Srivastava: Interpretable and Informative Explanations of Outcomes. PVLDB 8(1): 61-72 (2014)

[7] K. El Gebaly, L. Golab, J. Lin: Portable In-Browser Data Cube Exploration. IDEA workshop at KDD 2017: 35-39

[8] L. Golab, H. Karloff, F. Korn, D. Srivastava, B. Yu: On Generating Near-optimal Tableaux for Conditional Functional Dependencies. *PVLDB* 1(1): 376-390 (2008)

[9] L. Golab, F. Korn, D. Srivastava: Efficient and Effective Analysis of Data Quality using Pattern Tableaux. *IEEE Data Eng. Bull.* 34(3): 26-33 (2011)

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten: The WEKA Data Mining Software: An Update. *SIGKDD Explorations*, 11(1): 10-18 (2009)

[11] H. Lakkaraju, S. H. Bach, J. Leskovec: Interpretable Decision Sets: A Joint Framework for Description and Prediction. KDD 2016: 1675-1684

[12] S. Roy, L. Orr, D. Suciu: Explaining Query Answers with Explanation-Ready Databases. *PVLDB* 9(4): 348–359 (2015)

[13] S. Sarawagi: User-Cognizant Multidimensional Analysis. *VLDB Journal* 10(2-3): 224-239 (2001)

[14] X. Wang, X. L. Dong, A. Meliou: Data X-Ray: A Diagnostic Tool for Data Errors. SIGMOD 2015: 1231–1245

Ontology-Based Natural Language Query Interfaces for Data Exploration

Chuan Lei, Fatma Özcan, Abdul Quamar, Ashish Mittal,
Jaydeep Sen, Diptikalyan Saha, Karthik Sankaranarayanan
IBM Research - AI

1 Introduction

Enterprises are creating domain-specific knowledge bases by curating and integrating all their business data, structured, unstructured and semi-structured, and using them in enterprise applications to derive better business decisions. One distinct characteristic of these enterprise knowledge bases, compared to the open-domain general purpose knowledge bases like DBpedia [16] and Freebase [6], is their deep domain specialization. This deep domain understanding empowers many applications in various domains, such as health care and finance.

Exploring such knowledge bases, and operational data stores requires different querying capabilities. In addition to search, these databases also require very precise structured queries, including aggregations, as well as complex graph queries to understand the various relationships between various entities of the domain. For example, in a financial knowledge base, users may want to find out "*which startups raised the most VC funding in the first quarter of 2017*"; a very precise query that is best expressed in SQL. The users may also want to find all possible relationships between two specific board members of these startups, a query which is naturally expressed as an all-paths graph query. It is important to note that general purpose knowledge bases could also benefit from different query capabilities, but in this paper we focus on domain-specific knowledge graphs and their query needs.

Instead of learning and using many complex query languages, one natural way to query the data in these cases is using natural language interfaces to explore the data. In fact, human interaction with technology through conversational services is making big strides in many application domains in recent years [13]. Such interfaces are very desirable because they do not require the users to learn a complex query language, such as SQL, and the users do not need to know the exact schema of the data, or how it is stored.

There are several challenges in building a natural language interface to query data sets. The most difficult task is understanding the semantics of the query, hence the user intent. Early systems [3, 30] allowed only a set of keywords, which had very limited expressive power. There have been works to interpret the semantics of a full-blown English language query. These works in general try to disambiguate among the potentially multiple meanings of the words and their relationships. Some of these are machine-learning based [5, 24, 29] that require good training sets, which are hard to obtain. Others require user feedback [14, 17, 18]. However, excessive user interaction to resolve ambiguities can be detrimental to user experience.

In this paper, we describe a unique end-to-end ontology-based system for natural language querying over complex data sets. The system uses domain ontologies, which describe the semantic entities and their relationships, to reason about and capture user intent. To support multiple query types, the system provides a poly store

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

architecture, which includes a relational database, an inverted index document store, a JSON store, and a graph database. As a result, we support very precise SQL queries, document search queries, JSON queries, as well as graph queries.

Once the data is stored in the appropriate backends, we also need to provide the appropriate abstractions to query the data without knowing how data is stored and indexed in multiple data stores. We propose a unique two-stage approach: In the first stage, a natural language query (NLQ) is translated into an intermediate query language, called Ontology Query Language (OQL), over the domain ontology. In the second stage, each OQL query is translated into a backend query (e.g., SQL) by using the mapping between the ontology and the corresponding data schema of the backend.

In this two-stage approach, we propose an interpretation algorithm that leverages the rich semantic information available in the ontology, and produces a ranked list of interpretations for the input NLQ. This is inspired by the search paradigm, and minimizes the users' interaction for disambiguation. Using an ontology in the interpretation provides a stronger semantic basis for disambiguation compared to operating on a database schema.

OQL provides a powerful abstraction layer by encapsulating the details of the underlying physical data storage from the NLQ interpretation, and relieving the users from understanding what type of backend stores are utilized, and how the data is stored in them. The users only need to know the domain ontology, which defines the entities and their relationships. Our system understands the mappings of the various ontology concepts to the backend data stores as well as their corresponding schemas, and provides query translators from OQL to the target query language of the underlying system, providing physical independence.

The NLQ interpretation engine uses database data and synonyms to map the tokens of the textual query to various ontology elements like concepts, properties, and relations between concepts. Each token can map to multiple ontology elements. We produce an interpretation by selecting one such mapping for each token in the NLQ, resulting in multiple interpretations for a given NLQ. Each interpretation is then translated into an OQL query. The second step in the process translates OQL to the underlying target data store, using the ontology-to-schema mappings.

Conversational interfaces are the natural next step, which extends one-shot NLQ to a dialog between the system and the user, bringing the context into consideration, and allowing informed and better disambiguation. In this paper, after we describe our end-to-end NLQ system, we also discuss how to extend and adapt our ontology-based NLQ system for conversational services, and discuss the challenges involved.

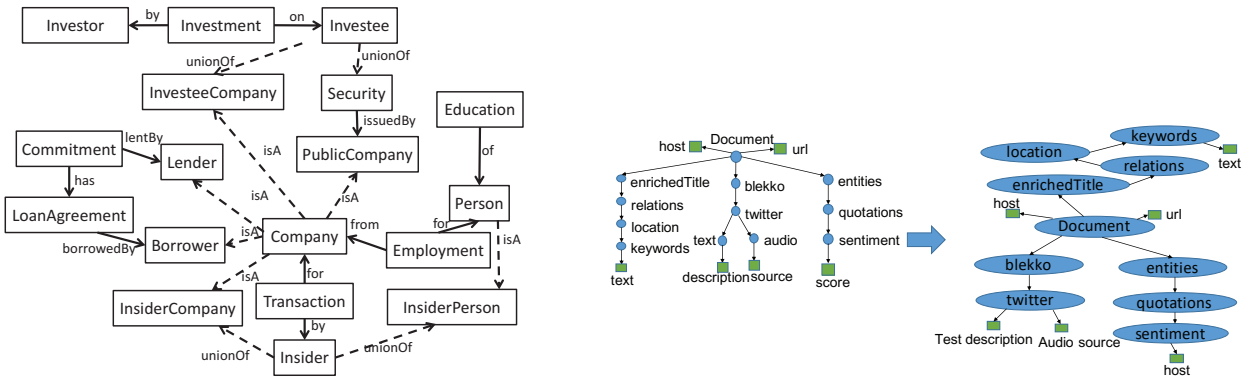
In the following, we first describe how we use ontologies, and the ontology query language (Sections 2 and 3), followed by the overall system description (Section 4). The translation index is used to capture the domain vocabulary and the domain taxonomies (Section 5). Section 6 discusses the details of the NLQ engine and Section 7 discusses the challenges in NLQ and conversational interfaces. We highlight some use cases where use our end-to-end system in Section 8, review related work in Section 9, and finally conclude in Section 10.

2 Ontology

2.1 Ontology Basics

The domain ontology is a core piece of technology, which is central to our system. It describes the domain and provides a structured entity-centric view of the data corresponding to the domain. Specifically, the ontology describes the entities relevant to the domain, the properties associated with the entities, and the relationships among different entities. We use OWL to describe our ontologies. It provides a very rich and expressive data model that can capture a variety of relationships between entities such as *functional*, *inheritance*, *unions*, etc. We use the domain ontology to just describe the schema of the domain capturing its meta-data. As such, in our proposed system, ontologies do not contain any instance data.

Figure 1(a) shows a small portion of an ontology that we use in a finance application [26, 8]. It includes concepts such as *Company*, *Person*, *Transaction*, *Loan agreement* etc. Each of these concepts are associated



(a) A semantic graph of partial finance ontology

(b) Ontology generation from schema

Figure 1: Sample Ontology and Ontology Generation

with a set of data properties: For example, the concept *Company* is associated with *name*, *stock symbols*, etc. Object properties represent the associations between the concepts. For example, *Securities* are *issued by* a *Public Company* where *issued by* is an object property. In this paper, we will use the extended version of this financial ontology to illustrate system functionality.

2.2 Generating Ontologies from Data

There are many cases where the ontology is provided, especially if the data is curated, but there are also cases where we only have the data without a corresponding ontology. To provide NLQ over such data sets, we provide techniques to infer the ontology, i.e., the concepts and their relationships relevant to the domain, from the underlying data and its schema.

Ontology generation from JSON data. This process involves the discovery of the domain ontology from all the JSON documents stored in a document store. This is a multi-step process. The first step involves the extraction of the schema tree from the nested JSON structure of each individual document. These individual schema trees are merged into a single schema, similar to the data guide generation process of Goldman et al. [11]. In the second step, we use the following rules to convert the schema into an ontology.

Path $A.b \Rightarrow$ Concept A, Property b of A

Path $A.B.c \Rightarrow$ Concept A, Concept B, Relation A to B, Property c of B

The ontology to physical schema mappings are an essential part of query translation, and are generated as part of ontology discovery. Figure 1(b) shows the schema that is generated by our techniques and the ontology generated from the schema, respectively.

Ontology generation from relational databases. Various ontology elements can be inferred from an RDBMS. *Table Inference.* Ontology supports two different types of relations - functional and ISA. The table inference is nontrivial as they depend on the Unique (Primary) Key and Foreign Key interactions, and quite often these keys are not specified in the database, especially when the database is created from raw data files. The steps are as follows:

- Identifying Unique Key and Foreign keys: For each table, we identify unique keys by comparing the count of the distinct values of the column and the total row count in the table. If they are identical, we assert a unique key constraint. Similarly, for foreign key, we check if the rows in the join of the two tables based on the selected columns are equal to the total rows of the referring table.

- Inferring ISA relation: Once all the unique keys and FKs have been derived, we find all the tables having a single column, which is both a unique key and the FK for the table. In such a case, we assert an ISA relation

between this table and the referred table. There might be spurious ISA relation if there are auto-incremented columns in two otherwise unrelated tables. To eliminate such cases, we check if the values in the column of the referencing table are consecutive integers. The rest of the tables are deemed functional.

- Inferring cardinality of functional relations: For inferring m:n relations, we find tables with exactly two columns, and both acting as foreign keys to different tables in the data base. If such a case occurs, the two tables that are being referred to are mapped in an m:n relation and the join table does not take part in any relation. We identify the relations with unique values in the FK and mark it as 1:1, and the rest as 1:n.

Concepts and Properties. Each table is mapped to a concept except the ones that form the join tables for the m:n relations, and all columns except the foreign keys in that table map to the data properties of that concept. The data type of the properties are also extracted from the data type of the table columns. The resulting ontology is stored in OWL2 [1] format.

3 Ontology Query Language

We propose OQL, a query language expressed against the domain ontology as an abstraction to query the data without knowing how data is stored and indexed in multiple data stores. OQL represents queries that operate on a set of concepts and relationships in an ontology. OQL is inspired by SQL and its object-oriented extensions, and it provides very similar constructs, including SELECT, FROM, WHERE, HAVING, GROUP BY and ORDER BY clauses. As such, OQL can express sophisticated queries that include aggregations, unions, nested sub queries, document, fielded search over document stores, as well as JSON paths, etc.

OQL allows several types of predicates in the WHERE clause including: (1) predicates used to compare concept properties with constant values (or sets of values), (2) predicates used to express joins between concepts, (3) predicates that use binary operations such as *MATCH* for full text search or fielded search over documents, and (4) predicates that use path expressions to specify the application of a predicate along a particular path over the domain ontology. In general OQL is composable and allows arbitrary level of nesting in the SELECT, FROM, WHERE and HAVING clauses.

Example 1: Show me all loans taken by Caterpillar by lender. This query joins the following concepts in the ontology: *lender*, *borrower*, *commitment* and *loanAgreement*. In this query, Caterpillar is the borrower. The condition *br = la→borrowedBy* indicates that the borrower instance should be reachable by following the *borrowedBy* relationship from the *LoanAgreement* concept.

```
SELECT   Sum(la.amount)
FROM     Lender ld, Borrower br, Commitment c, LoanAgreement la
WHERE    br.name IN ('Caterpillar', 'Caterpillar Inc.') AND ld = c→lentBy
         AND la = c→has AND br = la→borrowedBy
GROUP BY ld.name
```

Example 2: The following query applies a fielded search using the *MATCH* binary operation for a person whose name contains the word *Adam* and has held the position/title of *President* in a company.

```
SELECT  oPerson.name, oEmployment.position, oEmployment.title, oCompany.name
FROM    Person oPerson, Employment oEmployment, Company oCompany
WHERE   oPerson.name MATCH ('Adam') AND oEmployment.title='PRESIDENT'
        AND oEmployment→for=oPerson AND oEmployment→from = oCompany
```

4 System Architecture

Figure 2 shows the overall system architecture. It serves a wide variety of applications including NLQ Service, Conversational Services that enable chat bots, and other applications that use programmatic APIs. These ap-

plications use OQL queries to explore and analyze the underlying data, stored in one of the supported backend stores.

The *OQL query compiler* and the *OQL query translators* together translate the OQL query into the target back-end query using the ontology to physical schema mappings. By decoupling the semantic domain schema from the actual data storage and organization, our system enables independent optimization of each layer, and allows the applications to reason at the semantic level of domain entities and their relationships.

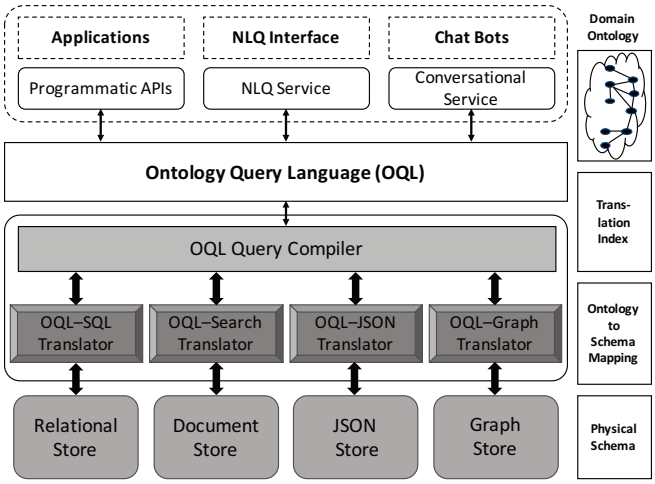


Figure 2: System Architecture

OQL Query Compiler. The OQL query compiler includes an OQL query parser, and a set of query translators one for each back-end. The query parser takes an OQL query as input and generates a logical representation of the query in the form of a Query Graph Model (QGM) [22]. Using QGM as the logical representation of the query defers the choice of the actual physical execution plan to the underlying data store that would be responsible for the execution of the query. The query compiler identifies which backend store contains the data needed by the query, and routes the query to that target. It may also generate a multi-store execution plan, which is optimized to minimize data movement between data stores.

OQL Query Translation. The query fragments expressed over the ontology are translated into appropriate back-end queries to be executed against the physical schema of the stored data. We have developed several query translators (one per type of back-end store) for this purpose. Query translation requires appropriate schema mappings that map concepts and relations represented in the domain ontology to appropriate schema objects in the target physical schema.

Query translation also needs to handle *union* and *inheritance* relationships, and translate path traversals to a series of join conditions. Depending on the physical data layout these are translated to appropriate operations supported by the back-end stores. For document oriented stores the translator needs to make distinctions between fielded and document oriented search. For querying JSON data it needs to understand where to apply predicates along specific paths in the nested schema.

5 Translation Index

Translation Index (TI) is a standalone component that captures the domain vocabulary, providing data and metadata indexing for data values, and for concepts, properties, and relations, respectively. TI helps the NLQ engine to identify the entities in the query, and their corresponding ontology properties.

TI captures both internal as well as external vocabularies. For the internal ones, TI is populated from the underlying data stores during an offline initialization phase. For example, if the input query contains the token "Alibaba", TI captures that "Alibaba" is a data value for the name column in the Company table in the relational back-end store. Note that "Alibaba" can be mapped to multiple ontology elements (e.g., InvestorCompany or Lender), and TI captures all of them. TI provides powerful and flexible matching by using semantic variant generation schemes. Essentially, for the data values indexed in TI, we not only index the actual values (e.g., distinct values appearing in Company.name), but also variants of those distinct values. TI leverages semantic variant generators (SVGs) for several common types, including person and company names, among others. For example, given an input string "Alibaba Inc", the company name SVG produces the following list of variants:

{“Alibaba”, “Alibaba Inc”, “Alibaba Inc.”, “Alibaba Incorporated”}. This allows the NLQ engine to formulate the queries by using any of the indexed variants of a data value (e.g., “Alibaba” vs. “Alibaba Inc”), or use these indexed variants in an IN (list) predicate.

Internal vocabulary is not always sufficient. Suppose “renal impairment” is a data value in the back-end store. If the query term is “kidney disease”, TI would not be able to return “renal impairment” even though “renal impairment” is a more specific term for “kidney disease”. To address this issue, TI also leverages external knowledge sources, such as standard taxonomies, domain lexicons, dictionaries, etc. To continue our example of searching for “kidney disease”, TI uses SNOMED [2], a systematically organized taxonomy of medical terms, to connect “kidney disease” with “renal impairment” via the *ISA* relationship in SNOMED. Hence with more domain knowledge, the semantically related matches can be found by TI, improving recall for the query.

6 Natural Language Querying

In this section, we describe how the *NLQ Engine* takes a natural language query (NLQ) and interprets it over the domain ontology to produce an OQL query. The transformation of NLQ to OQL query happens in multiple phases. We describe each of the phases with a running example from the finance domain as captured in Figure 1(a) for natural language query: *“Show me total loan amount given by Citibank to Caterpillar in 2016”*.

Evidence Generation. In the very first step, *NLQ Engine* tries to identify all the different mentions of ontology elements in the NLQ. So the algorithm first iterates over all the word tokens in the NLQ and collects *evidences* of one or more ontology elements (*concept*, *relation*, and *property*) which have been referenced in the input NLQ.

In general, a token can match multiple elements in the ontology. For example, the token “Citibank” is mapped to *Lender.name* and *Borrower.name* (considering Citibank may have acted as a borrower as well as a lender) and the phrase “loan amount” is mapped to *LoanCommitment.amount*. The phrase “given by” gets mapped to the *lent by* relation in the ontology. “Caterpillar” is also mapped to *Lender.name* and *Borrower.name*. The temporal expression “in 2016” gets mapped to *LoanCommitment.year*.

An evidence can be of three types: 1) a *metadata* evidence is generated by matching the *synonyms* associated with the ontology elements (e.g., “loan amount”), 2) a *data-value* evidence is generated by looking up a token in the *Translation Index* (e.g., “Citibank”,”Caterpillar”), and 3) a *typed* evidence is generated by recognizers such as temporal expressions that recognize date/time related expressions and maps them to date/timestamp typed properties in the ontology. For example, “in 2016” is detected and mapped to *LoanCommitment.year*. The evidence for a token can either be *metadata* or *data-value*, but not both. Note that if a token matches a data-value corresponding to an inherited property of a child concept then it should also match with the same property of the parent. For example, if “Citibank” data-value matches *Lender.Name* then it should also match *Company.Name*.

The metadata matching uses synonyms associated with the ontology elements. The NLQ engine relies on the given set of synonyms associated with ontology elements. It also uses publicly available synonym databases such as PPDB [10] to create additional synonyms for the given set of terms.

Interpretation Generation. Note that, only one element from the evidence set of each token corresponds to the correct query. In this phase, the NLQ engine tries all such combination of elements from each evidence set. Each such combination, called *selection set*, is used to generate an *interpretation*, which is represented as a subgraph in the semantic graph connecting one evidence for each token respecting ontology-related constraints. This semantically grounds the words in the natural language to specific meanings by referring to elements in the semantic graph. Connecting these referred elements produces a unique interpretation for the given natural language query based on the ontology semantics.

For each selection set, a subtree, called Interpretation Tree, is computed (if possible) which uniquely identifies the relationship paths among the evidences in the selected set. It is computed by connecting all the elements in the selected set in the semantic graph and satisfying two constraints. One such constraint is the *relationship*

constraint which asserts that between evidence mapping "loan amount", "given by", and "Citibank" there should be a path in the interpretation tree connecting them in order. When "Citibank" is mapped to *Borrower* then this constraint will not be satisfied. The other constraint is called inheritance constraint. Inheritance constraint invalidates those interpretation trees which have a parent (or union) concept as a chosen element and it connects to a property or a relation which belongs to one of its child (or member) concepts in the ontology.

Note that many interpretations can arise since there can be many possible subtrees connecting elements of a selected set in addition to many possible selected sets. We use a Steiner-tree-based algorithm to generate a single interpretation of minimal size from a selected set. The rationale of computing Steiner trees to derive the minimal sized connected tree follows from the intuition that among the different possible subtrees, the most compact one has the most direct and straight forward meaning in the ontology and thus is the most likely intent of the user query. Following the same intuition, it also employs a ranking criteria to choose one or more interpretations across all selected sets depending on the number of edges present in the minimal Steiner tree computed from a specific selected set. Fewer edges in the Steiner tree results into higher rank of the interpretation. For example, the top ranked interpretation as found from selected set is *ITree* = {(*LoanCommitment→lent by→Lender*), (*LoanCommitment→has→LoanAgreement*), (*LoanAgreement→borrowed by→Borrower*)}. Note that, the ranking can result in multiple interpretations with the top rank score, if there are multiple minimal Steiner trees with the same number of edges. In that case NLQ will provide multiple answers to the user as obtained from multiple interpretations for the question.

OQL Query Generation. As described in the previous section, our interpretation algorithm produces a ranked list of interpretations for a given NLQ. The goal of the Ontology Query Builder is to represent each interpretation in that list as an OQL query. In the final step, the constituent clauses of an OQL query corresponding to a given interpretation tree (ITree) and a selected set (SS) are generated as follows.

- FROM Clause. The FROM clause is formed by the evidence concepts in the interpretation graph. The path between the root node to other evidence nodes in the tree denote the join path between the concepts. Thus, the FROM clause for the example *ITree* will be *FROM Lender LenderObject, Borrower BorrowerObject, LoanCommitment LoanCommitmentObject*.

- WHERE clause. The WHERE clauses are generated based on the filter conditions specified. For example, to generate equality condition in the WHERE clause, data value evidences are used to determine the equality between the ontology property and the original value in the data value evidence. We uses a large vocabulary (e.g., more than, less than, etc.) for creating expressions. We also employ a grammar to recognize time-related expressions. For example, the WHERE clause corresponding to the example *ITree* will be *WHERE LenderObject.name = 'Citibank' AND BorrowerObject.name = 'Caterpillar' and LoanCommitmentObject.year = 2016*.

- SELECT clause. The SELECT clause contains a list of ontology properties which are categorized as aggregation properties and display properties depending on whether an aggregation function is applied to them. The OQL query generation algorithm identifies explicit references to aggregation functions (e.g., SUM/total) in the NLQ by employing a lexicon of terms corresponding to the common aggregation functions. In this example, the expression, *SUM(LoanCommitmentObject.amount))*, will be created. The display/non-aggregation properties of the SELECT clause are generated by finding ontology properties which appear after the head phrases such as "Show me", "Tell me" and not associated with another keyword such as "per" which can signify GROUPBY clause.

- GROUPBY clause. It specifies an ordered list of ontology properties that the user wishes to group the results by. A group by operation is recognized by presence of specific keywords like "by" , "for each" etc. followed by an ontology property.

- ORDERBY clause. The ORDERBY clause specifies an ordered list of ontology properties (or aggregations over properties) that the user wishes to order the results by. Like group by, order by operation is also recognized by presence of specific keywords like "order by", "sort by" followed by a property in the ontology.

7 Challenges and Vision

In this section we will discuss some of the common challenges encountered by the *NLQ Engine*. These are still open problems where active research is being done. Then, we also present our vision on how *NLQ Engine* can be extended to create a *Conversational System* that can resolve ambiguity via user interaction, which is also one of the core challenges in *NLQ Engine*.

7.1 Ongoing Work

Spurious Mapping in Evidence Generation. Consider an example query in "Show me companies Caterpillar has borrowed money from". In this example, the token "money" happens to match a person name in the underlying data store and therefore associates the candidate *Person.Name* property to that token. Such mapping is an instance of spurious mapping. Detecting and handling spurious mapping remains a challenge in developing robust interpretation generation algorithms. A possible way to do this is to use the contextual knowledge from the input query to identify the query is asking about "Loans" and "Loans" does not involve "Person" as per the ontology model. This knowledge can be utilized to prune out such spurious mappings from being included in the interpretation process.

Nested Query Handling. Although OQL grammar supports multiple levels of nesting, detecting an NLQ which is to be translated to a nested OQL query and subsequently producing correct OQLs for it remains an open challenge. A part of the difficulty lies in taking clues from the natural language for detecting different cases of possible nesting. A harder challenge is to identify the sub-queries and the correct join condition to stitch the results of the sub-queries. Although this still remains an open problem, here we try to provide a high-level overview of our ongoing effort to address this problem. We employ a nested query detector, which builds on different lexicon rules to detect different cases of nesting. For each of the different cases, a reasoning engine can work towards building different sub-queries. Consider an example question as "show me all IBM transactions with value more than average selling price", this can be detected as a nested query by the presence of comparator phrases like "more than" in the input query and a comparison against another aggregation value like "average selling price". A possible way to handle this query is to decompose the input NLQ text into two sentences across the comparator phrase i.e., LHS sentence as "show me all transactions with value" which corresponds to the outer OQL query, and RHS sentence "average selling price" which corresponds to the inner OQL query. The inner query, in this example, represents a predicate on a property referenced in the outer query i.e., Transaction.amount. At the next step, the Ontology-driven Interpretations Generator can be invoked on the LHS and RHS sentences, and then the Ontology Query Builder can generate two OQL queries corresponding to the two sentences. The reasoning engine can figure out the correct join between the inner and outer query result. For example, if they are aggregation comparisons on both sides, then the inner query can be incorporated in the HAVING clause of the outer query, or else if it is a property value comparison, like in this example question, then a simple property value comparison in the WHERE clause is performed between outer query and inner query result; e.g., $Transaction.amount > Average(Price)$.

7.2 Extending NLQ for Conversation

Typical natural language interfaces are stateless and so they are oblivious to the history of the questions being asked in the past. Users, while asking questions to the NLQ interfaces, are latently aware of the context and ask questions keeping that in mind. Consider the following interaction in the finance domain - (i) *What is the*

loan amount given by Citibank to Caterpillar in each year? (ii) *Who are its lenders?* Here, the second question is dependent on the context of the first question. These are natural interactions for users. The conversational interface enables users to automatically exploit the latent semantic context of the conversation, thereby making it simpler to express complex intents in a natural, piece-wise manner.

There are several challenges in supporting such conversational interfaces. These challenges arise in interpreting the contextual questions in the presence of the context. In the subsequent subsections, we list the challenges and how they are supported via extending the Natural Language Querying described in section 6.

Co-reference Resolution. In a natural language dialog, users often use co-references to refer to the entities in the previously asked question. These co-references are typically pronouns, anaphora, cataphora, and sometimes split antecedents. There are many state of the art solution for co-reference resolution, but all of them are trained on general purpose news data [15, 19, 21, 23]. They fail to identify the subtleties of any specific domain because of the lack of training data.

For example a follow up query - "*Who are its lenders?*" to the base query - "*What is the loan amount given by Citibank to Caterpillar?*", contains a pronoun *its* that needs to be resolved. There are two candidate to *its* viz. (i) *Citibank* (the lender) and (ii) *Caterpillar* (the borrower). Any state of the art system will fail the subtle difference between the companies, which are taking different roles in the domain. To circumvent this weakness, we use signals from the Ontology to resolve co-references. In this particular case, from the ontology described in Figure 1, a borrower can only have lenders (determined through path between the concepts), and hence correct resolution will associate the co-referent *its* to Caterpillar. Thus, using domain specific ontology augments existing techniques and yields improved accuracy.

Ellipsis Resolution. Another challenge in conversational interface is to support ellipsis (Non Sentential Utterances) [9]. Ellipsis are partial questions that modify the previously asked question. A sample interaction involving ellipsis is as follows - (i) "*What is the loan amount given by Citibank to Caterpillar?*", (ii) "*in 2015?*" Here, the second question cannot be interpreted on its own and modifies the previously asked question. In this particular case, a WHERE clause on the loan period should be added to the previous query.

To resolve the ellipsis, we use the Ontology to find the annotations (select, where) in the follow up query and then use these annotations to apply transformations to the previous query. For the example described above, the annotations in the base query are - *loan amount* is part of SELECT clause, *Citibank* and *Caterpillar* are part of WHERE clause. This process is described in detail in section 6. For the follow up query, we identify the temporal condition as a WHERE clause predicate, and since there is no existing temporal clause in the previous query, we add it to the previous query.

Disambiguation. Often times in natural language queries, users employ terms that are ambiguous, e.g., *Southwest* as (i) *Southwest Airlines* or (ii) *Southwest Securities*. A conversational interface can help resolving these ambiguities via interaction in natural language. Sometimes these ambiguities can be automatically resolved if the users have mentioned the unambiguous choices in the previous questions. If no such clarity is available from the context, the interaction proceeds by asking the user to clarify.

Context Management. Since a conversational interface needs to maintain the context of the previously asked questions and answers, the interface can no longer be stateless. Context manager is responsible for updating the context as conversation proceeds. The conversation is initialized with an empty context. When user asks a question, all the information related to that question(including annotations, answers, etc.) are packaged as a context object and pushed to the context, which is represented as a stack. If additional user input is required for disambiguation, a new context state is created specifically for the disambiguation. Once an answer to disambiguation question is detected, the entire disambiguation object is removed from the context, and the last question that caused the ambiguity is interpreted with the received answer.

8 Use Cases

The NLQ system has been applied in various domains. Here we describe the some of the important use-cases.
Finance. We instantiated the NLQ system for the finance domain by taking data from 5 years of SEC and FDIC filings which amounts to 1.5 TB of raw data. We created an ontology combining information from two standard finance-domain ontology called FIBO (Financial Industry Business Ontology) and FRO (Financial Regulation Ontology) with the help of domain experts. We processed the data such that it adheres to our ontology and instantiated our NLQ system on top of it. The main challenge of this use case was to overcome ambiguities across various entities e.g., there are 202 different entity matches just for the token *IBM*, spreading across 6 different roles such as Company, Subsidiary, Lender, Startup, Investor, and Investee. Our interpretation engine showed great precision and recall [26] in handling such cases.
Healthcare. We also worked on a healthcare use case by building a natural language conversation service to enable the pharmacists, physicians, and nurses to easily access various drug information. In particular, information pertaining to drug classes, drug synonyms, side effects, adverse effects, symptoms, general doses is accessed from a relational database through a conversation interface. The database is described using an ontology, and the conversation service intents and entities are identified with the guidance from the ontology. The interactive nature of the conversation service allows the system to resolve ambiguities and narrow down on specific answers, e.g., if a user asks *"what is the side effect of Aspirin"* then the system can ask *"for an adult or for a child"* to give specific information.
SAP-ERP. In this use-case, we extended the NLQ technology to interact with the SAP-ERP system. Because of the large size of SAP-ERP domain (100K tables and 5M relations), this use-case posed multiple challenges regarding automatically creating the ontology, capturing the domain vocabulary and populating the translation index, as well as the query translator, which required adaptations to the SAP SQL dialect. SAP-ERP has around 63 domains and 2K sub-domains, and we have created a goal-directed algorithm to create and instantiate NLQ for each specific domain/sub-domain, leveraging meta-data information from SAP's data dictionary tables. We have instantiated our system for different domains like Sales and Distribution, FICO (FInancial accounting) and Security Administration. The details of our solution are described in [27].
Weather. We have instantiated the NLQ system on weather data such that users can ask various weather-related questions in natural language. Two important features of this use-case were 1) handling time-related expressions, and 2) domain rules related to weather and time. Consider the question - *"Will it be hot tomorrow afternoon?"* In this case, we have to map "hot" to a temperature using a domain rule (hot : temperate $\geqslant$ 40-degree celsius) and afternoon implies 12 pm to 6 pm. We also have to enable interactive disambiguation for location entities (e.g., Columbus could mean Columbus, OH or Columbus, GA).

9 Related Work

In the context of data management, natural language interfaces for relational databases (NLIDB) have been studied for several decades [4, 28]. As noted in [4, 17], early NLIDBs were based on grammars designed particularly for a specific database, thus making it difficult to incorporate other databases. The most recent work in the area is the NaLIR system [4], which operates directly on the database schema as opposed to our two-stage approach that provides physical independence, and exploits the powerful semantics of the ontology. Moreover, NaLIR mainly relies on user interaction to find the correct interpretation for ambiguous NLQs. Our system, on the other hand, provides an interpretation ranking mechanism that almost eliminates user interaction. Similar to NaLIR, Nauda [14] is an early interactive NLIDB. However, its main focus is to provide additional useful information to user, more than what is explicitly asked in the question. The PRECISE system [25] defines a subset of NLQs as semantically tractable, and generates SQL queries only for these NLQs. Similar to NaLIR, PRECISE operates directly on the database schema and thus cannot exploit the rich semantics of the ontology.

Moreover, PRECISE is not able to rank the NLQ interpretations but returns all of them to the user. We believe that interpretation ranking provides a better user experience.

Ontology-driven data access systems [7, 26, 12, 20] capture the domain semantics and provide a standard description of the domain for applications to use. Some of these works [7, 26] either focus on specific application domains or offer ontology-based data access through description logic or semantic mappings that associate queries with the underlying data stores to the elements in the ontology. However, they either fail to support multiple query types in a poly store architecture or lack a natural language interface to explore the data stored in multiple backends.

10 Conclusion

In this paper, we described our ontology-based end-to-end NLQ system to explore databases and knowledge bases. We argued the importance of using domain ontologies and entity-based reasoning in interpreting natural language queries. We illustrated our experience of instantiating the system on top of various domains. As future work, we plan to extend NLQ to support more complicated nested queries. We are also investigating how to bootstrap a conversation service using an ontology, and leverage our NLQ stack for domain specific chat bots.

References

[1] Owl 2 web ontology language document overview. `https://www.w3.org/TR/owl2-overview/`. Accessed: 2018-06-01.

[2] Snomed ct. `https://www.snomed.org/snomed-ct/what-is-snomed-ct`. Accessed: 2018-07-02.

[3] B. Aditya, G. Bhalotia, S. Chakrabarti, A. Hulgeri, C. Nakhe, P. Parag, and S. Sudarshan. Banks: Browsing and keyword searching in relational databases. In *VLDB*. VLDB Endowment, 2002.

[4] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. Natural language interfaces to databases–an introduction. *Natural language engineering*, 1(01):29–81, 1995.

[5] J. Berant et al. Semantic Parsing on Freebase from Question-Answer Pairs. In *EMNLP*, pages 1533–1544, 2013.

[6] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *ACM SIGMOD*, 2008.

[7] D. Calvanese, G. De Giacomo, D. Lembo, et al. The mastro system for ontology-based data access. *Semant. web*, pages 43–53, 2011.

[8] B. et.al. Creation and interaction with large-scale domain-specific knowledge bases. *Proc. VLDB Endow.*, 10(12):1965–1968, Aug. 2017.

[9] R. Fernández and J. Ginzburg. Non-sentential utterances: Grammar and dialogue dynamics in corpus annotation. In *Computational linguistics*, pages 1–7. Association for Computational Linguistics, 2002.

[10] J. Ganitkevitch, B. V. Durme, and C. Callison-burch. Ppdb: The paraphrase database. In *In HLT-NAACL 2013*, 2013.

[11] R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. VLDB '97, pages 436–445, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[12] E. Kharlamov, T. P. Mailis, K. Bereta, et al. A semantic approach to polystores. In *IEEE Big Data*, 2016.

[13] K. Kolhe, E. Perez, and C. Sawicki. Voice assistants (va): Competitive overview, 2018.

[14] D. Küpper, M. Storbel, and D. Rösner. NAUDA: A Cooperative Natural Language Interface to Relational Databases. In *ACM SIGMOD*, 1993.

[15] H. Lee, Y. Peirsman, A. Chang, N. Chambers, M. Surdeanu, and D. Jurafsky. Stanford's multi-pass sieve coreference resolution system at the conll-2011 shared task. In *Computational Natural Language Learning*, 2011.

[16] J. Lehmann, R. Isele, M. Jakob, et al. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 2015.

[17] F. Li and H. V. Jagadish. Constructing an Interactive Natural Language Interface for Relational Databases. *Proc. VLDB Endow.*, 8(1):73–84, 2014.

[18] Y. Li, H. Yang, and H. V. Jagadish. NaLIX: An Interactive Natural Language Interface for Querying XML. In *ACM SIGMOD*, 2005.

[19] C. D. Manning, M. Surdeanu, J. Bauer, J. R. Finkel, S. Bethard, and D. McClosky. The stanford corenlp natural language processing toolkit. In *ACL (System Demonstrations)*, pages 55–60, 2014.

[20] J. McHugh, P. E. Cuddihy, J. W. Williams, et al. Integrated access to big data polystores through a knowledge-driven framework. In *IEEE Big Data, 2017*, 2017.

[21] V. Ng. Supervised noun phrase coreference research: The first fifteen years. In *Association for computational linguistics*, pages 1396–1411. Association for Computational Linguistics, 2010.

[22] H. Pirahesh, J. M. Hellerstein, and W. Hasan. Extensible/rule based query rewrite optimization in starburst. In *ACM SIGMOD*, pages 39–48, 1992.

[23] H. Poon and P. Domingos. Joint unsupervised coreference resolution with markov logic. In *Empirical methods in natural language processing*, pages 650–659. Association for Computational Linguistics, 2008.

[24] A.-M. Popescu et al. Modern Natural Language Interfaces to Databases: Composing Statistical Parsing with Semantic Tractability. In *COLING*, 2004.

[25] A.-M. Popescu, O. Etzioni, and H. Kautz. Towards a Theory of Natural Language Interfaces to Databases. In *IUI*, 2003.

[26] D. Saha, A. Floratou, K. Sankaranarayanan, U. F. Minhas, A. R. Mittal, and F. Özcan. Athena: An ontology-driven system for natural language querying over relational data stores. *Proc. VLDB Endow.*, 9(12):1209–1220, Aug. 2016.

[27] D. Saha, N. Gantayat, S. Mani, and B. Mitchell. Natural language querying in sap-erp platform. In *ESEC/FSE 2017*, pages 878–883, New York, NY, USA, 2017. ACM.

[28] U. Shafique and H. Qaiser. A comprehensive study on natural language processing and natural language interface to databases. *International Journal of Innovation and Scientific Research*, 9(2):297–306, 2014.

[29] L. R. Tang and R. J. Mooney. Using Multiple Clause Constructors in Inductive Logic Programming for Semantic Parsing. In *ECML*. 2001.

[30] S. Tata and G. M. Lohman. SQAK: Doing More with Keywords. In *ACM SIGMOD*, 2008.

The Periodic Table of Data Structures

Stratos Idreos Kostas Zoumpatianos Manos Athanassoulis Niv Dayan Brian Hentschel
Michael S. Kester Demi Guo Lukas Maas Wilson Qin Abdul Wasay Yiyou Sun

Harvard University

Abstract

We describe the vision of being able to reason about the design space of data structures. We break this down into two questions: 1) Can we know all data structures that is possible to design? 2) Can we compute the performance of arbitrary designs on a given hardware and workload without having to implement the design or even access the target hardware? If those challenges are possible, then an array of exciting opportunities would become feasible such as interactive what-if design to improve the productivity of data systems researchers and engineers, and informed decision making in industrial settings with regards to critical hardware/workload/data structure design issues. Then, even fully automated discovery of new data structure designs becomes possible. Furthermore, the structure of the design space itself provides numerous insights and opportunities such as the existence of design continuums that can lead to data systems with deep adaptivity, and a new understanding of the possible performance tradeoffs. Given the universal presence of data structures at the very core of any data-driven field across all sciences and industries, reasoning about their design can have significant benefits, making it more feasible (easier, faster and cheaper) to adopt tailored state-of-the-art storage solutions. And this effect is going to become increasingly more critical as data keeps growing, hardware keeps changing and more applications/fields realize the transformative power and potential of data analytics. This paper presents this vision and surveys first steps that demonstrate its feasibility.

1 Automating Data Structure Design

Data structures is how we store and access data. A data structure design consists of 1) the data organization, 2) an optional index, and 3) the algorithms that support basic operations (e.g., put, get, update). All **algorithms** deal with data and most often their design starts by defining a data structure that will minimize computation and data movement. Effectively, the possible ways to design an algorithm (and whole data systems) are restricted by the data structure choices [31, 76, 77, 47, 24, 8, 1, 46]. For example, we can only utilize an optimized sorted search algorithm if the data is sorted and if we can maintain the data efficiently in such a state. In turn, a **data analytics** pipeline is effectively a complex collection of such algorithms that deal with increasingly growing amounts of data. Overall, data structures are one of the most fundamental components of computer science and broadly data science; they are at the core of all subfields, including data analytics and the tools and infrastructure needed to perform analytics, ranging from data systems (relational, NoSQL, graph), compilers, and networks, to storage for machine learning models, and practically any ad-hoc program that deals with data.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

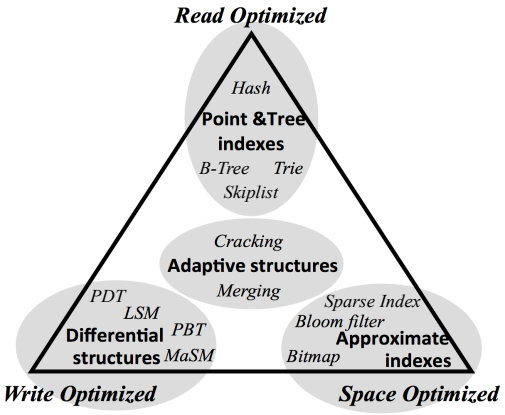


Figure 1: Each design compromises between read, update and memory amplification.

There is no perfect data structure design; each design is a compromise among the fundamental performance trade-offs [8]: read, update and memory amplification. This is depicted visually in Figure 1. In this way, to get good performance, the design of a data structure has to be tailored to the specific data, and query workload of the target application in addition to the underlying hardware where the data structure is expected to live.

A continuous challenge. Since the early days of computer science, the research community publishes 40-80 data structures per year. The pace has increased over the last decade because of the growth of data, the ever-increasing numbers of new data-driven applications, more fields moving to a computational paradigm where data analysis becomes critical, and hardware changing faster with characteristics that require a complete redesign of data structures and algorithms. Overall, with workloads and hardware changing frequently and rapidly, designing new data structures becomes a continuous challenge. Furthermore, for the several past decades, the trend in terms of hardware evolution is that computation becomes relatively faster than data movement; this makes data structure design even more critical as the way we store data dictates how much data an algorithm has to move.

The vast design space slows progress. There are so many different ways to design a data structure and so many moving targets that it has become a notoriously hard problem; it takes several months even for experts to design and optimize a new structure. Most often, the initial data structure decisions for a data system remain intact; it is too complex to redesign a data structure, predict what its impact would be as workload and hardware keep changing, implement it, and integrate it. For example, it is extremely hard to decide whether to acquire new hardware and how to adjust the core data structure design to it. Similarly, given the quick pace of adding new features in modern applications, the workload also changes rapidly; it is extremely hard to predict the potential performance impact and what possible design changes are best to allow for graceful adaptation by investing and timing engineering efforts accordingly. In the same lines, it is very hard to know when and why a data system would "break", i.e., when performance would drop below an acceptable threshold, so that we can invest in protecting against those scenarios and avoid prolonged degradation in performance or even down-time.

For fields without computer science/engineering expertise that move towards a data-driven paradigm, these low-level choices are impossible to make, and the only viable solution is using suboptimal off-the-shelf designs or spending more money in hiring experts. Modern data analytics scenarios and data science pipelines have many of those characteristics, i.e., domain experts with limited expertise on data structure design are confronted with dynamic, ever changing scenarios to explore and understand data. The likelihood that a single off-the-self data structure fits such arbitrary and sometimes unpredictable data access patterns is close to zero, yet data movement is the primary bottleneck as data grows and we do need more data to extract value out of data analytics.

The source of the problem is that, **there is no good way to predict the performance impact** of new workload, new hardware and new data structure design choices before fully implementing and testing them, and **there is no good way to be aware of all the possible designs**. We make three critical observations.

1. Each data structure design can be described as a set of design concepts. These are all low-level decisions that go into a given design such as using partitioning, pointers or direct addressing.

2. Each new data structure design can be classified in one of three ways; it is a set that includes: a) a new combination of existing design concepts, b) a new tuning of existing concepts or c) a new design concept.

3. By now we have invented so many fundamental design concepts such that most new designs fall within the first two categories, i.e., they are combinations or tunings of existing design ideas.

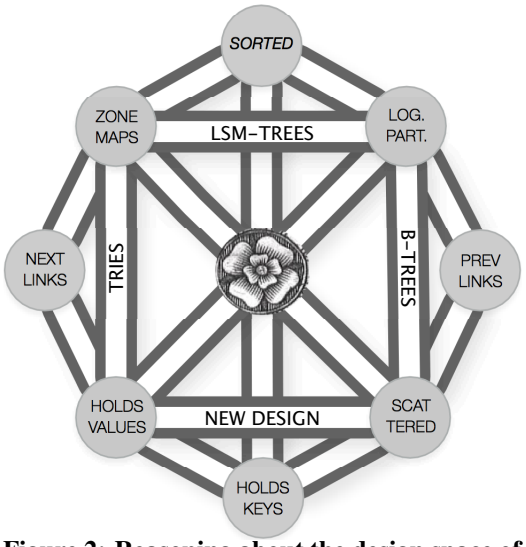


Figure 2: Reasoning about the design space of data structures by understanding its first principles and how they combine.

The design space structure is key. With most designs being combinations of existing design ideas, we argue that it is a great time to study in depth the design space itself to discover its structure and properties. Let us make an analogy to make it more clear why this is an important goal long term. Consider the periodic table of elements in chemistry. It organized existing elements into a universal structure, a design space formed based on their properties and their components, i.e., the atomic number and electron configuration. The crucial observation is that this design space not only classifies and explains connections among existing elements, but its structure (i.e., the gaps) provide hints about possible elements that had not been discovered yet, and it even helped argue about their expected properties. Even a hundred years after the original periodic table was conceived, scientists kept doing research driven by the gaps in the table, and kept discovering elements whose existence and properties had been predicted purely by the structure of the design space. Another example comes from computer science research on cache consistency algorithms in client-server environments. Mike Franklin's Ph.D. work led to a design space that mapped existing algorithms based on the common fundamental design decisions under which we could describe all these algorithms [21]. Similarly to the periodic table, once the structure was in place, we could classify existing algorithms, but also we could see additional algorithms that had not been invented yet; these new algorithms are formed as combinations of the design principles and the design space allows us to even argue about their properties.

Art vs. science. Just because a specific design is a combination of existing design concepts, it does not mean that it is easy to be conceived manually. In other words, when the number of options and possible design combinations is so big, it is hard even for experts to "see" the optimal solutions even when these solutions consist of existing concepts only. This is why many argue that research on algorithms and data structures is a form of art, i.e., driven by inspiration. Our goal is to accelerate this process and bring more structure to it so that inspiration is complemented by intuitive and interactive tools.

We set out to discover the first principles of data structure design and map the design space that they form. Our goal is to be able to **reason about their combinations**, tunings, and performance properties they bring. The high-level concept is depicted in Figure 2 using a small sample of principles for brevity. Effectively, the principles and their structure form a "grammar" with which we can describe any data structure in a systematic way, including designs that have not been discovered yet but they can be "calculated" from the design space given constraints such as the expected workload, and hardware environment. Figure 2 is inspired by a similar figure in the Ph.D. thesis of Gottfried Leibniz who in the 1600s envisioned an engine that calculates physical laws from a small set of primitives [48]. In the rest of this paper, we describe the process of discovering the **first principles of design**, we provide examples, and discuss the process of synthesizing arbitrary data structure designs. Similarly to the periodic table of elements in chemistry, the final overall structure, which we call **the periodic table of data structures**, provides insights and explanations on existing designs but it also points to exciting research opportunities. We then discuss how to automatically **synthesize the optimal algorithms** of the basic operations of a data structure based on the target workload and hardware using **an expert system and learned cost models**. Learned cost models represent fundamental access patterns (sorted search, random access, etc.) that can be combined into arbitrary algorithms. In addition, they allow computing the performance (response time) of a data structure design on a target workload and hardware without implementing and testing it. Finally, we discuss new opportunities that arise from the design space, i.e., discovering **design continuums** and making steps toward the ultimate goal of **automatic data structure design**.

2 The Quest for the First Principles of Data Structure Design

We define as a first principle a design concept that is not possible to break into more fine-grained concepts. For example, consider the design choice of linking two nodes of a data structure with a pointer. While there are potential tuning options (e.g., the size of the pointer), it is not possible to break this decision further; we either introduce a pointer or not.

Layout vs. access. We separate design principles that have to do with the layout of a data structure from those that have to do with how we access the data; layout drives the design of algorithms. Our intuition is that we can describe data structures solely by their layout, while the **access algorithms can be automatically derived** for a given set of layout decisions.

Thus, we focus on discovering, structuring and understanding the first principles of layout design. This is a trial and error process, i.e., there is no explicit algorithm. There is a **high-level recipe**, though. The recipe is about iterating between two processes: 1) decomposing existing data structures into first principles, and 2) stress testing the design space by attacking open problems through its structure. The second step is critical as many principles that were thought to be essential for specific subspaces of the design space can be optimized out, while new principles are being discovered for design decisions that remained fixed until now. The end goal is to develop a universal model that describes all structures as compositions of design principles. As more data structure instances are being considered, the set of first principles and the universal model change. This happens at a decreasing pace as by definition there are much fewer principles than data structure instances.

Arrays, B-Trees, and LSM-trees have been our primary playground for applying the above recipe as they cover a massive part of the overall design space. Arrays are practically part of all data structures (e.g., the nodes of a B-tree, the buckets of a hash-table). Arrays are also the fundamental storage model in modern analytical systems and large-scale cloud infrastructures where columnar storage is the leading paradigm. B-trees represent the evolution of tree-based structures, and they are the primary form of indexing on data systems for the past several decades to support read-intensive workloads. Finally, LSM-trees form the backbone of NoSQL data systems being a highly write-intensive design and a rather unexplored design space. Decomposing these three major classes of data structures and their variations into their first principles has led us to several surprising results that arise purely by the design space and the study of its structure.

For example, a study of access path selection in relational databases (i.e., dynamically choosing between a scan over an array and accessing a B-tree secondary index) shows that following the fixed selectivity threshold paradigm as was invented in the 1970s is suboptimal. After considering all design principles such as multi-core and SIMD parallelism, read and write patterns and costs, and shared reads, we showed that there should not be a fixed selectivity threshold for access path selection but rather a dynamic one that depends on the number of concurrent requests [39]. Similarly, the work on Smooth Scan shows that it is possible to combine these two fundamental access paths into a new hybrid one that combines their properties and requires no access path selection [12]. Finally, the work on Column Sketches showed that by considering the design principle of lossy compression, we can enable new performance properties for scans over an array that were not possible before, i.e., robust performance regardless of selectivity, data, and query properties [32].

Decomposing LSM-trees has revealed that all modern designs in industry are suboptimal due to fixing fundamental design principles to suboptimal decisions. For example, the work on Monkey shows that for a given memory budget the number of bits assigned to the bloom filter of each level (or run) should be exponentially smaller for bigger levels [18]. This allows for smaller false positive ratios for smaller levels and since each level contributes in the same way to the overall LSM-tree worst case false positive ratio, this tuning provides a drastic improvement that leads to orders of magnitude better performance. Similarly, the work on Dostoevsky shows that the merge policy should not be fixed across levels and that we can tune it in a different way for each level. This helps lower the bounds for writes without hurting reads and it also allows to tune the merge policy according to the desired low bounds for point reads, range reads and writes [19]. We have performed a similar process in other areas including bit vector structures [9] and structures for statistics computation [73].

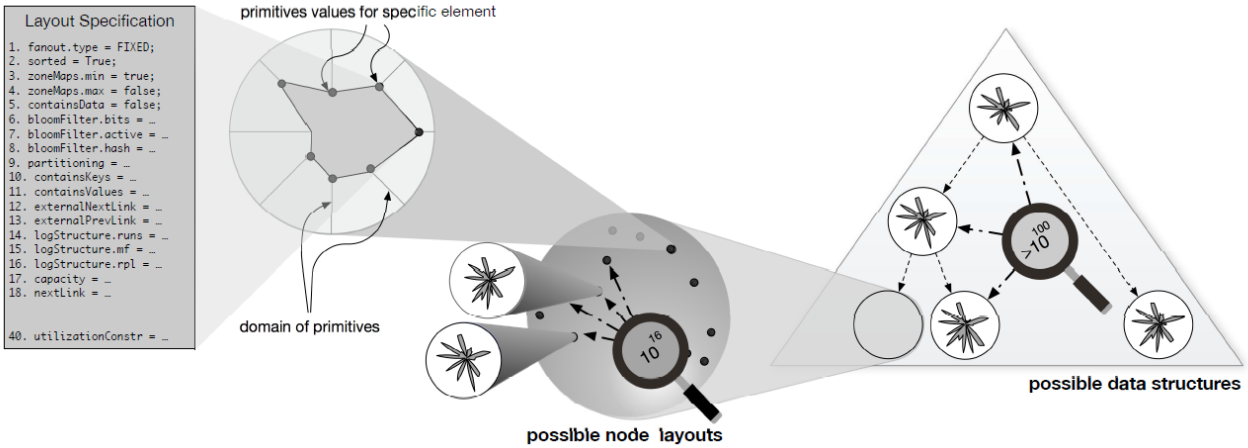


Figure 3: Synthesizing a vast number of data structure designs from first principles.

Examples of first principles include fundamental design concepts such as whether a given node of a structure contains keys or values. For example, an internal node of a B+tree does not contain any keys or values while a leaf node does. Furthermore, there are decisions that have to do with the physical layout. As an example, say that a node does contain keys and values; are these physically stored as key-value pairs (e.g., a row-based format) or as separate arrays (e.g., a columnar format)? A critical design principle is partitioning; does a node help break the domain into smaller pieces and, if yes, how exactly and what is the fanout? For example, we may use range partitioning, radix partitioning, temporal partitioning or any other function. If partitioning is enabled, then this means that each node generates sub-blocks; how do we access these sub-blocks? Through direct addressing or pointers? In addition, are the sub-blocks connected to each other with immediate links like a linked list or skip links like in a skip list? Further design principles have to do with filters. Do we have filters that help us access the sub-blocks of a node, e.g., bloom filters? And if we do, how many bits and how many hash functions should be used? Other types of filters are also possible such as zone maps. Furthermore, how are filters stored? Are they scattered across the sub-blocks or stored in a contiguous area? Then, there are design principles that have to do with how the design evolves outside the current node. For example, if the sub-blocks do not live inside the current node, then they form its children. Are these children stored scattered in memory, e.g., as the children of a B+tree node or they are stored in a contiguous block as the children of a cache-conscious B+tree node?

The overall model is based on iteratively describing one node of a data structure at a time. Figure 3 depicts a high-level view of this process (from left to right). Through the principles, we define data nodes as sets of design decisions. Their combinations craft the space of possible node descriptions. Then a data structure is described by connecting one or more node types and describing how they connect and potentially recurse. In practice, it is slightly more complex. With each set of principles, we describe a "logical block" of data that may be directly mapped onto an actual data structure node type, but it may also be a function we apply to the data. Most data structures in the literature consist of two logical blocks. For example, a B+Tree consists of two logical blocks, describing the internal and the leaf nodes respectively, while a hash table also consists of two logical blocks, describing the hash function and the hash buckets respectively. The initial set of layout principles covers the key-value model for read-only operations and appeared in the Data Calculator work [36]. The model allows the description of numerous well-known structures and their hybrids, including arrays, linked-lists, skip-lists, queues, hash-tables, binary trees and (cache-conscious) B-trees, tries, MassTree, and FAST.

What comes after crafting the design space is understanding and utilizing its structure. Given the sheer size of the design space, though, before we can do that effectively we need to be able to argue about any point in the space so we can quickly and automatically compare designs. Section 3 discusses cost synthesis which helps with this problem and then Section 4 describes existing results and opportunities in utilizing the design space.

3 Algorithm and Cost Synthesis from Learned Cost Models

To fully utilize the knowledge of the vast design space we need to be able to compare different designs in terms of the expected performance. Complexity analysis explains how the properties of a design scale with data but does not give a precise performance computation given a workload and hardware. On the other hand, full implementation and testing on the target workload and hardware provide the exact performance while the combination of both complexity analysis and implementation gives the complete picture. The problem is that with so many designs possible in the design space, it is not feasible to analyze, implement and test the numerous valid candidates. This is exactly the complex and time consuming process we are trying to avoid in the first place. A less explored option is building generalized hardware-conscious cost models [55, 76]. Similar to a full implementation, such models provide results very close to ground truth, and similar to complexity analysis they provide hints on how performance scales. However, building generalized models is nearly equally hard to a full implementation given the level of detail and hardware-conscious properties needed [39]. In addition, any change on hardware properties requires retuning the model. Overall, arguing formally about the performance of diverse data structures designs, especially as workload and hardware properties change, is a notoriously hard problem [55, 78, 72, 13, 76, 79, 39] and in our case we want to perform this task for numerous designs concurrently and at interactive response times.

In fact, before being able to argue about performance, we need to "complete" the design. The design space we crafted in the previous section is about the layout only, while a full design includes the access algorithms as well. We show in the Data Calculator [36] that algorithms can be built automatically by **synthesizing operations from fundamental access primitives** on blocks of data through a rule-based expert system. Rules are statements that map layouts to access patterns, e.g., if a data block is sorted we should use some kind of sorted search. However, the performance that we would get out of sorted search depends on 1) the exact algorithm (e.g., binary search, interpolation search), 2) the exact hardware, and 3) the exact engineering (e.g., using a new set of SIMD instructions). The rule base system gives us only a high-level sketch of the desired algorithm. Then, we synthesize the detailed algorithm and compute the cost via a hybrid of analytical models, implementation, and machine learning as combinations of a small set of fundamental access principles. We call these **learned cost models**. For each generic class of access pattern on a block of data (e.g., sorted search) there exist many instances of algorithms/implementations. These are short implementations that capture the exact access pattern. For example, for a binary search the implementation consists of an array of sorted integers where we perform a series of searches. During a training phase this code runs on the desired hardware and we learn a model as data scales (across the memory hierarchy). Each model captures the subtle performance details of diverse hardware settings and the exact engineering used to build the code for each access pattern. To make training easier, our models start out as analytical models since we know how these access patterns will likely behave.

Overall, after taking as input the layout of a design, a workload and a hardware, the Data Calculator goes step by step through the operations in the workload and for each operation it computes the expected state of the data structure at this point in the workload. This includes the number of nodes that exist and the number of elements in each node. For each operation it then computes the path that it should follow over the data structure by using the specification of its layout. This results in a set of nodes that should be visited. For each node, the expert system then provides the high level access pattern that should be used, e.g., probe a bloom filter, scan, sorted search. Once the high-level access pattern is known, the system then scans all possible models for particular implementations that exist in its library and chooses the one that is expected to behave the best on the target hardware and data size. Putting everything together, by repeating this process for all nodes in the path of an operation we can compute the total cost of this operation and by repeating this for all operations in a workload we get the overall cost. The exact process includes further low-level details to handle random access, caching and skew effects which are all handled by a combination of additional models and rules in the expert system for cost synthesis. As a first step, the Data Calculator work [36] shows how the above process takes place for bulk-loading, point, and range queries.

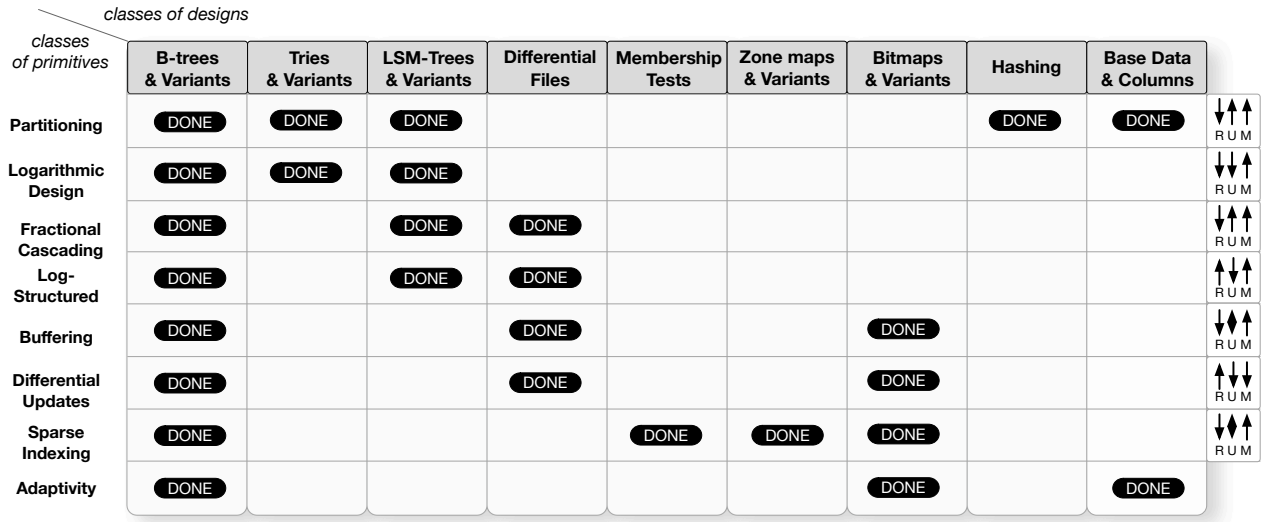


Figure 4: The periodic table of data structures: a vast unexplored design space.

4 The Periodic Table of Data Structures

In the same way that the arithmetic calculator can be used in any application where numbers are necessary (e.g., from complex scientific problems to restaurant bills), we expect that the Data Calculator will be useful in any application where data is essential. Here we describe a sample of exciting results, applications, and opportunities that have to do both with the structure of the design space, and with the ability to understand and navigate it.

4.1 Structure, design guides, gaps, and facts

Figure 4 depicts our first effort to unify known classes of data structures under a unified model. The table groups data structures and design primitives into classes. Every cell indicates whether any such principle has been applied to a category of data structures (i.e., a relevant paper has been published). The observation is that most of the cells are in fact empty. While this does not necessarily mean that these combinations have not been studied before, and it does not even suggest that they will all be useful for modern applications and hardware, it is still surprising. Also since this is a "zoomed out" table it loses some detail for purposes of visualization and thus even if a cell is marked as done, this does not mean that further work is not possible. By counting the exact possible designs from the detailed design space presented in the Data Calculator work [36], we find that there are at least 10^{32} two-node structures (and 10^{48} three-node ones). This is more possible designs than stars on the sky (10^{24}). We have barely published a couple of thousand of designs in the last sixty years, and at a rate of a hundred designs a year (as indicated by data from DBLP) we are moving very slowly.

Furthermore, the structure of the design space helps to predict how specific choices may affect the properties of a design with respect to read, update, and memory amplification. This is shown on the right-hand side of the table on Figure 4. Such hints can be used as a design guide by engineers and researchers (and even auto design algorithms) to accelerate design space exploration during the design process.

4.2 Interactive Design to Accelerate Engineering and Research

When combining the design space knowledge with the ability to do algorithm and cost synthesis we are presented with a unique and new opportunity for interactive design. That is, enabling researchers and engineers to quickly iterate over different designs without having to implement, test and even access the target hardware. For example, one can give as input a high-level specification of a data structure design, a sample workload, and the target hardware. The Data Calculator can then compute the performance we would get if we were to

implement this design and test it on this workload and hardware [36]. This computation takes a few seconds to complete, allowing researchers and engineers to **quickly iterate over: 1) multiple designs** to decide which one to spend time implementing or how specific changes may affect performance in an existing design; **2) multiple workloads** to inspect how envisioned changes in the workload (e.g., due to new application features) may affect the performance; **3) multiple hardware environments** to decide which is the best hardware for placement in a large cluster or to drive equipment purchase decisions by knowing the expected outcome on performance given the current design and workload. In our most recent results, such computations take in the order of 20 seconds for 1 Million data entries and 100 queries for a diverse class of structures including B-trees, Tries, and Hash-tables [36]. At the same time, training for typical modern hardware takes in the order of 50-100 seconds, and it is a one time process; once it is done, one can try out any workload and design on this hardware.

When it comes to interactive design a crucial question is that of **extensibility**. What if one wants to test a new access pattern that is not included in the Data Calculator? For example, say a researcher comes up with a new way to perform sorted search and wants to know the expected performance impact on their B-tree design. The new idea may be an algorithmic one or even a new hardware-conscious way of implementing an existing high-level algorithm, e.g., using a new set of SIMD instructions. All the researcher has to do is add a new benchmark under the sorted search family of access patterns and train it for the desired hardware. Then this becomes permanently part of the library of access patterns which has two positive side-effects: 1) the researcher can now test and refine their design with and without the new access pattern in a variety of workloads and hardware settings before initiating a complex implementation phase, and 2) the Data Calculator can now consider the new access pattern for any data structure design that includes sorted data (e.g., the buckets of a hash-table); in this way, future queries on the Data Calculator by this or other researchers in the same team can benefit by an expanded design space. On the other hand, extending the design space of layout principles requires expansion of the universal model and the associated rules for algorithm and cost synthesis. This is not a task that can be done interactively by users; it is the ongoing research effort to develop the theory behind such tools.

4.3 Design Continuums and Self-designing Systems

We define as design continuum a part of the design space of data structures that can be seen as a unified space because it is defined by the same set of first principles. Thus, there is a path between any two designs in this space by tuning the same set of principles. While tuning has always been possible and critical for data structures (e.g., tuning the fanout of a B-tree), the new opportunity here is that by structuring and understanding the design space, we can see such continuums across classes of data structures that are traditionally considered fundamentally different. For example, in our recent key-value store work we showed that there is a design continuum between logs, sorted arrays and LSM-trees [18, 19]. At the one end of the continuum exists a log, i.e., a write optimized structure where we can simply append new pages without having to maintain any order. Reads of course are slow as they require a full scan. On the other end of the spectrum exists a sorted array where we can quickly search with a binary search but writes require moving on average half the data. In between these two extremes exist LSM-trees and variations which give a balance between read and write costs. We showed that we can tune the shape of the structure **anywhere between a log and a sorted array** by tuning the merge policy of an LSM-tree. Specifically, an infinite size ratio and a tiering policy turn an LSM-tree into a log, while when using leveling and again infinite size ratio we get a sorted array. As we give different values to the size ratio and vary the merge policy for different levels, we get data structure designs which occupy points within the design continuum, can be seen as LSM-tree hybrids, and allow for custom read/write ratios and resource management [18, 19].

The discovery of design continuums opens the door for a new form of "deep adaptivity" by being able to "transition" between different designs. We term such systems **self-designing systems**, i.e., systems that can drastically morph their shape at run-time. Contrary to existing work on topics such as adaptive indexing which allows on-the-fly data adaptation, self-designing systems can transition across designs that are perceived as

fundamentally different (e.g., from log to LSM-tree to sorted array [18, 19]) and thus enabling a system to assume a much larger set of performance properties.

4.4 Automated Design: Computing Instead of Inventing

The ultimate dream has always been fully automated design [71, 76, 13]. We redefine this challenge here to include hardware; given a hardware and a workload, can we compute the best data structure design? The design space gives us a clear set of possible designs. In turn, algorithm and cost synthesis give us the ability to rank designs for a given workload and hardware. Together these two features allow the design of algorithms that traverse the design space, turning data structure design into a search problem. Early results have demonstrated that dynamic programming algorithms, searching part of the design space only, can automatically design custom data structures that fit the workload exactly [36]. What is even more intriguing is that these designs are new, i.e., they have not been published in the research literature, yet they stem from the design space as combinations of existing design principles and they match workloads better than existing state-of-the-art human-made designs. In another example, we showed that the search space can also be traversed via genetic algorithms, automatically designing column-store/row-store hybrids [35]. Further usages of the search capability include **completing partial designs**, and **inspecting existing designs** in terms of how far away they are from the optimal [36]. In addition, it is not necessary that the ultimate design should always be the target. For example, returning the top ten designs allows engineering teams to make decisions based on performance as well as implementation complexity, maintainability and other desirable properties.

The grant challenge is building algorithms that can traverse the whole design space. Given the exponential nature of the problem, we have started experimenting with reinforcement learning and Bayesian optimization with early results indicating that they can search a larger part of the design space within the same time budget.

5 Inspiration and Related Work

Our work is inspired by several lines of work across many fields of computer science. John Ousterhout's project Magic in the area of computer architecture allows for quick verification of transistor designs so that engineers can easily test multiple designs [58]. Leland Wilkinson's "grammar of graphics" provides structure and formulation on the massive universe of possible graphics one can design [74]. Mike Franklin's Ph.D. thesis explores the possible client-server architecture designs using caching based replication as the main design primitive and proposes a taxonomy that produced both published and unpublished (at the time) cache consistency algorithms [21]. Joe Hellerstein's work on Generalized Search Indexes [31, 6, 43, 42, 44, 45] makes it easy to design and test new data structures by providing templates that significantly minimize implementation time. S. Bing Yao's work on generalized cost models [76] for database organizations, and Stefan Manegold's work on generalized cost models tailored for the memory hierarchy [54] showed that it is possible to synthesize the costs of database operations from basic access patterns and based on hardware performance properties. Work on data representation synthesis in programming languages [63, 64, 17, 68, 66, 28, 29, 52, 69, 53] enables selection and synthesis of representations out of small sets of (3-5) existing data structures.

Work on tuning [37, 15] and adaptive systems is also relevant as conceptually any adaptive technique tunes along part of the design space. For example, work on hybrid data layouts and adaptive indexing automates selection of the right layout [7, 3, 27, 33, 20, 65, 4, 51, 18, 62, 25, 60, 80, 34, 38, 67]. Similarly works on tuning via experiments [10], learning [5], and tuning via machine learning [2, 30] can adapt parts of a design using feedback from tests. Further, the Data Calculator shares concepts with the stream of work on modular systems: in databases for easily adding data types [22, 23, 56, 57, 70] with minimal implementation effort, or plug and play features and whole system components with clean interfaces [50, 49, 16, 11, 14, 40, 61, 75], as well as in software engineering [59], computer architecture [58], and networks [41].

Research on new data structures keeps pushing the possible design space further. For example, an exciting idea is that of Learned Indexes which drastically expands the possible design space by proposing mixing machine learning models and traditional data structures; among other benefits, it enables new opportunities for succinct, yet performant, data structures [46]. Our work is complementary to such efforts as it is not toward expanding the design space, but rather toward mapping, understanding and navigating it. The Data Calculator can be seen as a step toward the Automatic Programmer challenge set by Jim Gray in his Turing award lecture [26], and as a step toward the "calculus of data structures" challenge set by Turing award winner Robert Tarjan [71]: *"What makes one data structure better than another for a certain application? The known results cry out for an underlying theory to explain them."*

6 Summary and Future Steps

We show that it is possible to argue about the design space of data structures. By discovering the first principles of the design of data structures and putting them in a universal model, we study their combinations and their impact on performance. We show that it is possible to accelerate research and decision making concerning data structure design, hardware, and workload by being able to quickly compute the performance impact of a vast number of designs; several orders of magnitude more designs than what has been published during the last six decades. Critical future steps include 1) expanding the supported design space (e.g., updates, concurrency, graphs, spatial), 2) supporting scalable and fast cost synthesis to allow for self-designing systems that change their shape at run-time, and 3) navigating the data structure design space by modeling it as a search, optimization and learning problem.

References

[1] D. J. Abadi, P. Boncz, S. Harizopoulos, S. Idreos, and S. Madden. The Design and Implementation of Modern Column-Oriented Database Systems. *Foundations and Trends in Databases*, 2013.

[2] D. V. Aken, A. Pavlo, G. Gordon, and B. Zhang. Automatic Database Management System Tuning Through Large-scale Machine Learning. *SIGMOD*, 2017.

[3] I. Alagiannis, S. Idreos, and A. Ailamaki. H2O: A Hands-free Adaptive Store. *SIGMOD*, 2014.

[4] V. Alvarez, F. Schuhknecht, J. Dittrich, and S. Richter. Main Memory Adaptive Indexing for Multi-Core Systems. *DAMON*, 2014.

[5] M. Anderson, D. Antenucci, V. Bittorf, M. Burgess, M. Cafarella, A. Kumar, F. Niu, Y. Park, C. Ré, and C. Zhang. Brainwash: A Data System for Feature Engineering. *CIDR*, 2013.

[6] P. Aoki. Generalizing "Search" in Generalized Search Trees (Extended Abstract). *ICDE*, 1998.

[7] J. Arulraj, A. Pavlo, and P. Menon. Bridging the Archipelago between Row-Stores and Column-Stores for Hybrid Workloads. *SIGMOD*, 2016.

[8] M. Athanassoulis, M. Kester, L. Maas, R. Stoica, S. Idreos, A. Ailamaki, and M. Callaghan. Designing Access Methods: The RUM Conjecture. *EDBT*, 2016.

[9] M. Athanassoulis, Z. Yan, and S. Idreos. UpBit: Scalable In-Memory Updatable Bitmap Indexing. *SIGMOD*, 2016.

[10] S. Babu, N. Borisov, S. Duan, H. Herodotou, and V. Thummala. Automated Experiment-Driven Management of (Database) Systems. *HotOS*, 2009.

[11] D. Batory, J. Barnett, J. Garza, K. Smith, K. Tsukuda, B. Twichell, and T. Wise. GENESIS: An Extensible Database Management System. *Transactions on Software Engineering*, 1988.

[12] R. Borovica-Gajic, S. Idreos, A. Ailamaki, M. Zukowski, and C. Fraser. Smooth Scan: Statistics-Oblivious Access Paths. *ICDE*, 2015.

[13] A. Cardenas. Evaluation and selection of file organization - A model and system. *Commun. ACM*, 1973.

[14] M. Carey and D. DeWitt. An Overview of the EXODUS Project. *Data Eng. Bul.*, 1987.

[15] S. Chaudhuri, V. Narasayya. An Efficient Cost-Driven Index Selection Tool for Microsoft SQL Server. *VLDB*, 1997.

[16] S. Chaudhuri and G. Weikum. Rethinking Database System Architecture: Towards a Self-Tuning RISC-Style Database System. *VLDB*, 2000.

[17] D. Cohen and N. Campbell. Automating relational operations on data structures. *Software*, 1993.

[18] N. Dayan, M. Athanassoulis, and S. Idreos. Monkey: Optimal Navigable Key-Value Store. *SIGMOD*, 2017.

[19] N. Dayan and S. Idreos. Dostoevsky: Better Space-Time Trade-Offs for LSM-Tree Based Key-Value Stores via Adaptive Removal of Superfluous Merging. *SIGMOD*, 2018.

[20] J. Dittrich and A. Jindal. Towards a One Size Fits All Database Architecture. *CIDR*, 2011.

[21] M. J. Franklin. Caching and Memory Management in Client-Server Database Systems. *Ph.D. Univ. of Wisconsin-Madison*, 1993.

[22] D. Goldhirsch and J. Orenstein. Extensibility in the PROBE Database System. *Data Eng. Bul.*, 1987.

[23] G. Graefe. Volcano - An Extensible and Parallel Query Evaluation System. *TKDE*, 1994.

[24] G. Graefe. Modern B-Tree Techniques. *Foundations and Trends in Databases*, 3(4), 2011.

[25] G. Graefe, F. Halim, S. Idreos, H. Kuno, S. Manegold. Concurrency control for adaptive indexing. *PVLDB*, 2012.

[26] J. Gray. What Next? A Few Remaining Problems in Information Technology. *SIGMOD Dig. Symp. Coll.*, 2000.

[27] R. Hankins and J. Patel. Data Morphing: An Adaptive, Cache-Conscious Storage Technique. *VLDB*, 2003.

[28] P. Hawkins, A. Aiken, K. Fisher, M. Rinard, and M. Sagiv. Data Representation Synthesis. *PLDI*, 2011.

[29] P. Hawkins, A. Aiken, K. Fisher, M. Rinard, and M. Sagiv. Concurrent data representation synthesis. *PLDI*, 2012.

[30] M. Heimel, M. Kiefer, and V. Markl. Self-Tuning, GPU-Accelerated Kernel Density Models for Multidimensional Selectivity Estimation. *SIGMOD*, 2015.

[31] J. Hellerstein, J. Naughton, and A. Pfeffer. Generalized Search Trees for Database Systems. *VLDB*, 1995.

[32] B. Hentschel, M. Kester, and S. Idreos. Column Sketches: A Scan Accelerator for Rapid and Robust Predicate Evaluation. *SIGMOD*, 2018.

[33] S. Idreos, M. Kersten, and S. Manegold. Database Cracking. *CIDR*, 2007.

[34] S. Idreos, M. Kersten, and S. Manegold. Self-organizing Tuple Reconstruction in Column-Stores. *SIGMOD*, 2009.

[35] S. Idreos, L. Maas, and M. Kester. Evolutionary data systems. *CoRR*, abs/1706.05714, 2017.

[36] S. Idreos, K. Zoumpatianos, B. Hentschel, M. Kester, and D. Guo. The Data Calculator: Data Structure Design and Cost Synthesis from First Principles and Learned Cost Models. *SIGMOD*, 2018.

[37] Y. Ioannidis and E. Wong. Query Optimization by Simulated Annealing. *SIGMOD*, 1987.

[38] O. Kennedy and L. Ziarek. Just-In-Time Data Structures. *CIDR*, 2015.

[39] M. Kester, M. Athanassoulis, and S. Idreos. Access Path Selection in Main-Memory Optimized Data Systems: Should I Scan or Should I Probe? *SIGMOD*, 2017.

[40] Y. Klonatos, C. Koch, T. Rompf, H. Chafi. Building Efficient Query Engines in a High-Level Language. *PVLDB'14*

[41] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek. The Click Modular Router. *TOCS*, 2000.

[42] M. Kornacker. High-Performance Extensible Indexing. *VLDB*, 1999.

[43] M. Kornacker, C. Mohan, and J. Hellerstein. Concurrency and Recovery in Generalized Search Trees. *SIGMOD'97*

[44] M. Kornacker, M. Shah, and J. Hellerstein. amdb: An Access Method Debugging Tool. *SIGMOD*, 1998.

[45] M. Kornacker, M. Shah, and J. Hellerstein. Amdb: A Design Tool for Access Methods. *Data Eng. Bul.*, 2003.

[46] T. Kraska, A. Beutel, E. Chi, J. Dean, and N. Polyzotis. The Case for Learned Index Structures. *SIGMOD*, 2018.

[47] T. Lehman, M. Carey. A Study of Index Structures for Main Memory Database Management Systems. *VLDB*, 1986.

[48] G. Leibniz. Dissertation on the art of combinations. *PhD Thesis, Leipzig University*, 1666.

[49] J. Levandoski, D. Lomet, S. Sengupta. LLAMA: A Cache/Storage Subsystem for Modern Hardware. *PVLDB*, 2013.

[50] J. Levandoski, D. Lomet, and S. Sengupta. The Bw-Tree: A B-tree for New Hardware Platforms. *ICDE*, 2013.

[51] Z. Liu and S. Idreos. Main Memory Adaptive Denormalization. *SIGMOD*, 2016.

[52] C. Loncaric, E. Torlak, and M. Ernst. Fast Synthesis of Fast Collections. *PLDI*, 2016.

[53] C. Loncaric, E. Torlak, and M. Ernst. Generalized Data Structure Synthesis. *ICSE*, 2018.

[54] S. Manegold. Understanding, modeling, and improving main-memory database performance. *Ph.D. UVA*, 2002.

[55] S. Manegold, P. Boncz, M. Kersten. Generic Database Cost Models for Hierarchical Memory Systems. *VLDB*, 2002.

[56] J. McPherson and H. Pirahesh. An Overview of Extensibility in Starburst. *Data Eng. Bul.*, 1987.

[57] S. Orborn. Extensible Databases and RAD. *Data Eng. Bul.*, 10(2), 1987.

[58] J. Ousterhout, G. Hamachi, R. Mayo, W. Scott, and G. Taylor. Magic: A VLSI Layout System. *DAC*, 1984.

[59] D. Parnas. Designing Software for Ease of Extension and Contraction. *TSE*, 1979.

[60] E. Petraki, S. Idreos, and S. Manegold. Holistic Indexing in Main-memory Column-stores. *SIGMOD*, 2015.

[61] H. Pirk, O. Moll, M. Zaharia, and S. Madden. Voodoo - A vector algebra for portable database performance on modern hardware. *PVLDB*, 2016.

[62] H. Pirk, E. Petraki, S. Idreos, S. Manegold, and M. Kersten. Database cracking: fancy scan, not poor man's sort! *DAMON*, 2014.

[63] E. Schonberg, J. Schwartz, and Sharir. Automatic data structure selection in SETL. *POPL*, 1979.

[64] E. Schonberg, J. Schwartz, and Sharir. An automatic technique for selection of data representations in SETL programs. *Transactions on Programming Languages and Systems*, 3(2), 1981.

[65] F. Schuhknecht, A. Jindal, and J. Dittrich. The Uncracked Pieces in Database Cracking. *PVLDB*, 2013.

[66] O. Shacham, M. Vechev, and E. Yahav. Chameleon: Adaptive Selection of Collections. *PLDI*, 2009.

[67] D. Sleator and R. Tarjan. Self-Adjusting Binary Search Trees. *Journal of the ACM*, 1985.

[68] Y. Smaragdakis and D. Batory. Distil: A transformation library for data structures. *DSL*, 1997.

[69] M. Steindorfer and J. Vinju. Towards a Software Product Line of Trie-Based Collections. *GPCE*, 2016.

[70] M. Stonebraker, J. Anton, and M. Hirohama. Extendability in POSTGRES. *Data Eng. Bul.*, 1987.

[71] R. Tarjan. Complexity of combinatorial algorithms. *SIAM Rev*, 1978.

[72] T. Teorey and K. Das. Application of an analytical model to evaluate storage structures. *SIGMOD*, 1976.

[73] A. Wasay, X. Wei, N. Dayan, S. Idreos. Data Canopy: Accelerating Exploratory Statistical Analysis. *SIGMOD'17*.

[74] L. Wilkinson. The grammar of graphics. *Springer*, 2005.

[75] H. Xing, S. Floratos, S. Blanas, S. Byna, Prabhat, K. Wu, and P. Brown. ArrayBridge: Interweaving declarative array processing in SciDB with imperative HDF5-based programs. *ICDE*, 2018.

[76] S. Yao. An Attribute Based Model for Database Access Cost Analysis. *TODS*, 1977.

[77] S. Yao and D. DeJong. Evaluation of Database Access Paths. *SIGMOD*, 1978.

[78] S. Yao and A. Merten. Selection of file organization using an analytic model. *VLDB*, 1975.

[79] M. Zhou. Generalizing database access methods. *Ph.D. Thesis. University of Waterloo*, 1999.

[80] K. Zoumpatianos, S. Idreos, and T. Palpanas. Indexing for interactive exploration of big data series. *SIGMOD*, 2014.

Impact Award Winner

I would like to thank TCDE and the committee for this great honor, and for an amazing event that we all experienced in Paris!

Very few things in life are entirely the work of just one person and this is no exception, so I must also pay tribute to the enormous contributions made by my students and colleagues without which, none of this would have been possible and I certainly wouldn't have gotten this award.

Awards like these tell us a lot about our pastime and the drive that has made this possible. And this drive has been possible because I was always surrounded by enthusiastic and inspiring people: my professors, my students and my colleagues. In fact, I have had great teachers and mentors throughout my whole career - undergrad, grad, postdoc, faculty - which is my inspiration for trying to be a good teacher and mentor myself. There are 3 pioneers that greatly influenced my career, Christos Papadimitriou in 1981 was the person to suggest that I should do a PhD in Databases, Phil Bersntein who was the first to mentor me at Harvard, and Mike Stonebraker for guiding me through a PhD and getting a real view of real DBMS group.

Back in the early 90s and after 9 years in the US, I felt that I had to go and give back to my home country, Greece, and have an impact on forming a data management community there. I am very happy to reflect now on the 20 years I devoted to this, giving me great pleasure, not only to see so many other colleagues came back to Greece as well, but also that we have collectively managed to set up and maintain a high caliber community.

In the last 6 years I live in Melbourne; well not so different as Melbourne is the third largest Greek city in the world after Athens and Thessaloniki! Leading a Data Science institute there presents a great challenge. I am trying to run an interdisciplinary unit working on research opportunities with astrophysicists to social scientists and hospitals to textile manufacturers. To be honest, what I enjoy most, apart from seeing how we can help all these communities, is how much I learn from processes and methodologies they use and thinking how these can influence data engineering. And I feel more and more that we nowadays play a much more important role than in the past.

Data is becoming the glue between disciplines and we have a responsibility in making interdisciplinary efforts work; as data management community we were always open to talk to other disciplines, as everybody presents us with interesting challenges. In that sense, if we gain their trust we can also help throw bridges among them. A personal experience: talking to astrophysicists and cancer researchers I could see the commonalities on image analysis research including data analysis and GPU based algorithms. Bringing them together to the table created a great interdisciplinary team and we look forward to a very successful collaboration. Data-driven solutions are in the core of problem solving nowadays and all sectors from anthropology to autonomous systems are looking towards data intelligence as the new paradigm for scientific discovery and innovation. There is a bright future for our field and we should be happy and feel the great responsibility to make this a successful future!

Timos Sellis
Swinburne University of Technology
Melbourne, Australia

Service Award Winner

I would like to express my sincere gratitude to the IEEE Technical Committee of Data Engineering for the 2018 TCDE Service award. It cites my long time involvement in the ICDE Conferences, clearly confirming Woody Allen observation that eighty percent of success is just showing up. Since receiving this award makes me officially an old-timer, I will reminisce briefly about the early years of the Data Engineering conference.

The conference was started by C.V. Ramamoorthy (who coined the term Data Engineering) and Gio Wiederhold. After the first five years in Los Angeles, the conference became well established and was ready to make the next step. The people who by then were running the conference (Mike Liu, Bruce Berra, and Benjamin Wah) made several decisions that significantly expanded its international reach and scope. The first, was to make the conference truly international by organizing it outside of US, in Europe, Asia, and Texas. The second, was to create of a companion event single topic, fully reviewed Workshops on Research Issues in Data Engineering (RIDE). The first RIDE in Kobe was organized by Yahiko Kambayashi and Ahmed Elmagarmid and started a successful series that lasted 15 years. By the turn of the century, the governance of ICDE was codified in the By-laws developed under the leadership of Erich Neuhold, which created an organizational structure that functions without major changes until today.

In the 35 years since its inception ICDE visited 4 continents, significantly expanded its thematic scope, and branched into the new areas that did not exist 30 years ago. It was always open to new ideas, not afraid to bring in new people and new exciting topics, never dull, and never much interested in what Mike Stonebraker called polishing a round ball.

There are many people who made great contributions to ICDE and to the field of Data Engineering. I would like to congratulate the leaders of TCDE for their excellent idea to recognize them through annual awards. I would also like to congratulate my fellow co-awardees. And still the best advice that I can give to our younger colleagues is: keep showing up.

Marek Rusinkiewicz
New Jersey Institute of Technology
Newark, New Jersey

Education Award Winner

It was extremely gratifying to be honored with the 2018 TCDE Education Award. While I've received various research awards over time, this is the first time I've been recognized for my educational endeavors. The truth is that most of my career I've been only an adequate teacher, certainly not among those who are truly gifted at the craft. I steadfastly taught my undergraduate and graduate classes to acceptable student ratings, and I dutifully updated the material periodically without being especially creative.

That all changed in the fall of 2011, when I jumped on the bandwagon with my faculty colleagues Sebastian Thrun (AI) and Andrew Ng (Machine Learning) to create one of Stanford's three inaugural free-to-the-world online courses, now known as MOOCs (Massive Open Online Courses). Even though I was chair of the Computer Science department at the time, and running a research group, I threw myself into the creation and delivery of the Databases MOOC with a passion for teaching I hadn't known that I possessed. The excitement and immense gratitude of tens of thousands of students was invigorating and inspirational. The Databases MOOC is still online in a self-serve modular fashion, and I continue to get a steady stream of thank-you's from students worldwide. Creating that course may turn out to be one of the broadest impact endeavors of my career.

Fast-forward to the fall of 2016 when it was time for a sabbatical. I wasn't especially interested a typical sabbatical that expands my own research horizons; I wanted to have direct impact on people. I combined my long-standing love of exotic travel with my MOOC experience to hatch a concept I jokingly call the MOIC, or Massive Open *In-Person* Course. With logistical help from ACM and many international students and colleagues, and with financial backing from ACM, VLDB, Google, and Stanford, I traveled to about 25 locations in 15 developing countries offering free short-courses on Big Data, workshops on "Design Thinking" (a wildly popular problem-solving methodology from Stanford's d.school), and roundtable discussions for women in technology. Once again I found myself deeply engaged in a teaching endeavor that reaped many rewards. Although my sabbatical was cut short when I was named Dean of Engineering, I endeavor to carve out at least two one-week periods per year for travel-teaching, and I've stuck with it so far. To date I've visited 18 countries and reached several thousand students in person across the globe.

I used the material I developed for my travel-teaching to launch a new Stanford class: Big Data Tools & Techniques for non-Computer Science majors. I'm not sure if it's the MOOC or MOIC experiences, the timeliness of the topic, the fact that (as Dean) I'm teaching for fun and not because I'm required to, but I'm enjoying my Stanford teaching more than I ever have before, and getting excellent student ratings to boot. Who would have guessed? Thanks again to TCDE for the recognition and honor.

Jennifer Widom
Stanford University
Palo Alto, CA

Rising Star Award co-Winner

I am humbled and honored to be selected as a co-recipient of the IEEE TCDE Rising Star award this year, and I am grateful to the TCDE community for this recognition.

Designing a database system for warehouse-scale computers

My research is motivated by the increasing use of warehouse-scale computers to analyze massive datasets quickly. Current examples include a scientist who post-processes simulation results on a high-performance computer or an enterprise that periodically rents 100,000 CPU cores in the cloud for $500 per hour to sift through IoT data.

These scenarios pose two challenges for database systems. The first challenge is the lack of *interoperability* with other analytical tools. Massive datasets often consist of images (arrays) that are stored in file formats like JPEG, PNG, FITS and HDF5. To analyze these datasets, users write code that directly manipulates data in files and leverages domain-specific libraries or deep learning frameworks such as TensorFlow. The second challenge is *scalability*, as foundational data processing operations do not scale to the unprecedented concurrency of warehouse-scale computers at the compute, networking and storage levels. Examples include data shuffling that triggers an inherently unscalable all-to-all communication pattern; hash-based joins that partition the inputs and do not allow communication and computation to fully overlap; parallel aggregations that are oblivious to congested links in non-uniform network topologies.

To address the interoperability concern, we are developing ArrayBridge [code.osu.edu/arraybridge], an open-source I/O library that brings advanced data management capabilities under an array-centric file format interface. ArrayBridge currently allows scientists to use SciDB, TensorFlow and HDF5-based code in the same analysis pipeline without converting between file formats. Under the hood, ArrayBridge controls I/O concurrency, data materialization and data placement to fully utilize warehouse-scale parallel file systems. ArrayBridge does not modify the array file format API, so it remains backwards compatibility with legacy applications. In other words, with ArrayBridge, scientists can simultaneously benefit from the I/O optimizations of a database system and directly manipulate data through the existing API when they need to.

To address the scalability concern, we are developing algorithms that use high-performance networks more efficiently. One example is an RDMA-aware data shuffling algorithm that directly converses with the network adapter in InfiniBand verbs and shuffles data up to $4\times$ faster than the RDMA-aware MVAPICH implementation of MPI. This impressive performance gain is achieved by carefully designing its data structures for concurrent operations in a multi-threaded setting and by using a connectionless, datagram-based transport layer that scales better but requires flow control and error detection in software. Another example is a parallel aggregation algorithm that predicts bottlenecks caused by link congestion and then leverages partition similarity to transfer less data over the congested link. We have prototyped both algorithms in Pythia [code.osu.edu/pythia], an open-source distributed query execution engine.

Looking ahead, the modern datacenter is becoming more heterogeneous with the introduction of FPGA nodes, many-core nodes, large memory nodes, GPU nodes and nodes with local NVMe-based storage. How to harness heterogeneous hardware for data analysis is an open research problem.

Spyros Blanas
Ohio State
Columbus, OH

Rising Star Award co-Winner

I am very honored to win this prestigious award. I would like to take this opportunity to thank TCDE committee members, award committees, and especially my nominators, for recognizing my research contributions to human-in-the-loop data analytics. I also want to thank my research collaborators and my graduate students. Without them, I would not be able to achieve what I achieved today. In particular, I want to thank five people, who have made a tremendous impact on my research career in the past ten years.

I did my Ph.D. in the Database Group at Tsinghua University from 2008 to 2013. Profs. Jianhua Feng and Guoliang Li were my Ph.D. supervisors. I want to say thanks to them from the bottom of my heart. They helped me grow from a fresh graduate, who knew little about SIGMOD, VLDB, or ICDE, to an independent researcher, who was able to independently publish a high-quality research paper in these top conferences. My Ph.D. thesis is entitled "CrowdER: Crowdsourcing Entity Resolution". It discussed how to build a hybrid human-machine Entity Resolution system, and validated that such a hybrid system can achieve both good efficiency and high accuracy compared to machine-only or human-only alternatives. Due to these contributions, the thesis won the China Computer Federation (CCF) Distinguished Dissertation Award (the most prestigious award for CS Ph.D. students in China).

I visited AMPLab at UC Berkeley for six months from 2011 to 2012, and then did a Postdoc for two years and a half from 2013 to 2015. I was so fortunate to have Prof. Michael Franklin as my supervisor and Prof. Tim Kraska as my mentor. They helped me grow from a fresh Ph.D. graduate, who had little experience to lead a large project, to a mature postdoc, who had known how to think big and do influential work. We started the SampleClean project with the vision of scaling up data cleaning and crowdsourcing to big data. I led a team of two Ph.D. students and two undergraduates to develop the SampleClean system. The system was incorporated into the BADS (Berkeley Data Analytics Stack). It helps users to extract value from dirty data, at significantly reduced time and cost.

I joined the School of Computing Science at Simon Fraser University (SFU) as an assistant professor in 2016. Prof. Jian Pei was my mentor. I feel deeply grateful to Jian. He helped me grow from a postdoc, who had no experience in setting up a research lab, to a junior faculty member, who has known how to launch a research lab and manage a group of students. My lab's mission is to *speed up data science*. We develop innovative technologies and open-source tools for data scientists such that they can turn raw data into actionable insights in a more efficient manner. We have already made a lot of progress on this mission.

- **Data Exploration.** We have built AQP++, an interactive analytics system that enables data scientists to query a large database interactively when they only have limited hardware.
- **Data Enrichment.** We have built Deeper, a data enrichment system that can help data scientists to reduce their time spent on data enrichment with Deep Web from hours to minutes.
- **Data Labeling.** We have built TARS, a label cleaning advisor that can provide valuable advice for data scientists when they need to train or/and test a model using noisy labels.

I demonstrated these systems in my Big Data Science course (`http://tiny.cc/sfu-ds`). Many students found them quite useful and tried to use them in their final projects.

In the past ten years, I feel lucky since I got so much support from so many people in the database community. A new journey has begun. In the next ten years, I hope that I can be the one who helps students grow to what they want to be and leads them to conduct database research that can make a big impact.

Jinnan Wang
Simon Fraser University
Vancouver, Canada

Call for Participation for TCDE Chair Election

Dear IEEE TCDE members,

The Chair of IEEE Computer Society Technical Committee on Data Engineering (TCDE) is elected for a two year period. The mandate of the current Chair, Xiaofang Zhou, expires at the end of this year and the election of a Chair for the period 2019-2020 has begun. A Nominating Committee consisting of Kyu-Young Whang (Chair) and Amr El Abbadi was formed on May 9th, 2018. After a thorough search, we have identified two highly qualified candidates, Thomas Risse and Erich Neuhold, who have confirmed their willingness to serve. Candidates bio-sketches and position statements appear in this issue of the Bulletin. The Nominating Committee encourages all members of TCDE who are IEEE CS members to vote to elect a new Chair. Voting will be open from Sept. 15th until Oct. 15th, 2018. Those who wish to become a TCDE member, please visit `http://computer.org/jointcde`

Kyu-Young Whang, Amr El Abbadi
IEEE TCDE Chair Nominating Committee

Thomas Risse Biography and Statement

Biography

Thomas Risse is head of Electronic Services at the University Library J. C. Senckenberg of the Goethe University Frankfurt. Before joining the University Library he was deputy managing director of the L3S Research Center and research group leader at L3S in Hannover, Germany. He received his diploma in Computer Science (1997) and his PhD in Computer Science (2006) from the Darmstadt University of Technology (TU Darmstadt), Germany. In 1998 he joined the Fraunhofer Institute for Integrated Publication System (FhG-IPSI) in Darmstadt where he worked in several European and industrial projects. From 2001-2006 he was the manager of the division for Intelligent Information Environments.

He was the coordinator or technical director of several European projects in the area of digital libraries and Web archives. He was also involved in the European research infrastructure project SoBigData on ethic-sensitive social mining. Thomas Risse serves regularly as program committee member or project reviewer. He published several papers at the relevant international conferences. In addition he is member of the Steering Committeeof theICDE conference (IEEE International Conference on Data Engineering), member of the Executive Committee of the IEEE Technical Committee on Data Engineering (TCDE), and member of the steering committee of the conference on Theory and Practice of Digital Libraries (TPDL). Thomas Risse was general co-chair of the 20th International Conference on Theory and Practice of Digital Libraries (TPDL 2016) and served as the local and financial chair of the 27th International Conference on Data Engineering (ICDE) in 2011 in Hannover.

Position statement

TCDE is a flagship international society for data management and database systems researchers and practitioners. If I get elected and have to honor to serve as the next TCDE Chair, I will continue the already well established collaboration between TCDE, ICDE Steering committee, VLDB Endowment and ACM SIGMOD to ensure that ICDE maintains its status as a top-tier international data management conference. In addition, the collaboration with other IEEE database related conference like BigData, ASONAM and MDM will be extended for the benefit of the community. I will work together with the TCDE executive committee to increase the visibility and to provide more benefits for TCDE members by increasing the number of local TCDE chapters and activities. In this context, I will work with the successful Data Engineering Bulletin editorial board on its evolution to keep it as a high value information source for all TCDE members. In order to give our members more recognition the TCDE awards will further promoted and developed. Since a healthy budget is important to ensure high quality conference and community activities, a stable income should be generated by increasing the number international high quality conferences.

Thomas Risse
Goethe University
Frankfurt, Germany

Erich Neuhold Biography and Statement

Biography

Erich Neuhold is Honorary Professor for Computer Science at the University of Vienna and Emeritus of the Technical University of Darmstadt. Until April 2005 he was Director of the Research Institute Fraunhofer IPSI and Professor at the Technical University of Darmstadt, Germany. His expertise includes semantic modeling of relational, unstructured and interoperable Databases, Digital Libraries, Archives and Web Information Stores like the Semantic Web and Linked Data and their applications. He has also been involved in security and privacy issues as they concern these fields. He has been Chair of TCDE 2002-2006 and is or has been on the Steering Committees of many conferences, among them ICDE and JCDL and has been involved in many congress, conference and workshop activities and in governmental and corporate bodies. He is a Fellow of IEEE and of the GI, Germany. He is a member of the T&C Board of IEEE-CS and Chair of the IEEE-CS CAC committee that is concerned about the quality of IEEE co-sponsored conferences. He has published four books and about 200 papers in Journals and Conferences.

Position statement

TCDE is a flagship international society for researchers and practitioners in data management systems and Web based information stores. I have been chairing it during the period 2002 to 2006 and it is a great honor that I have been asked to be a candidate to serve again. Should I have the honor to be elected as the next TCDE Chair, I will continue to work closely with the ICDE steering committee as well as the VLDB Endowment and ACM SIGMOD to ensure that ICDE maintains its status as a top-tier international database conference. I will work with the Data Engineering Bulletin editorial board to keep the bulletin the highly valued information source for all TCDE members and others in the field. I will try to follow the very successful former and current TCDE Chairs and work with the TCDE executive committee to continue to promote TCDE membership. I will also work with them to extend the number of high quality international conferences sponsored by TCDE and manage them carefully to ensure their quality and to build a healthy budget for TCDE to invest in the future of TCDE community. I also would like to strengthen our visibility by cooperating with more regional top quality conferences, workshops and seminars in different countries in order to provide more benefits and a region-focused collaboration platform for our members. We will continue to promote TCDE awards to give our members more recognition inside and outside of IEEE.

Erich Neuhold
University of Vienna
Vienna, Austria

Call for Papers

35th IEEE International Conference on Data Engineering

8-12 April 2019, Macau SAR, China

The annual IEEE International Conference on Data Engineering (ICDE) addresses research issues in designing, building, managing and evaluating advanced data-intensive systems and applications. It is a leading forum for researchers, practitioners, developers, and users to explore cutting-edge ideas and to exchange techniques, tools, and experiences. The 35th ICDE will take place at Macau, China, a beautiful seaside city where the old eastern and western cultures as well as the modern civilization are well integrated.

Topics of Interest

We encourage submissions of high quality original research contributions in the following areas. We also welcome any original contributions that may cross the boundaries among areas or point in other novel directions of interest to the database research community:

- Benchmarking, Performance Modelling, and Tuning
- Data Integration, Metadata Management, and Interoperability
- Data Mining and Knowledge Discovery
- Data Models, Semantics, Query languages
- Data Provenance, cleaning, curation
- Data Science
- Data Stream Systems and Sensor Networks
- Data Visualization and Interactive Data Exploration
- Database Privacy, Security, and Trust
- Distributed, Parallel and P2P Data Management
- Graphs, RDF, Web Data and Social Networks
- Database technology for machine learning
- Modern Hardware and In-Memory Database Systems
- Query Processing, Indexing, and Optimization
- Search and Information extraction
- Strings, Texts, and Keyword Search
- Temporal, Spatial, Mobile and Multimedia
- Uncertain, Probabilistic Databases
- Workflows, Scientific Data Management

Important Dates

For the first time in ICDE, ICDE2019 will have two rounds' submissions. All deadlines in the following are 11:59PM US PDT.

First Round:

Abstract submission due: May 25, 2018
Submission due: June 1, 2018
Notification to authors (Accept/Revise/Reject): August 10, 2018
Revisions due: September 7, 2018
Notification to authors (Accept/Reject): September 28, 2018
Camera-ready copy due: October 19, 2018

Second Round:

Abstract submission due: September 28, 2018
Submission due: October 5, 2018
Notification to authors (Accept/Revise/Reject): December 14th, 2018
Revisions due: January 11th, 2019
Notification to authors (Accept/Reject): February 1, 2019
Camera-ready copy due: February 22, 2019

General Co-Chairs
Christian S. Jensen, Aalborg University
Lionel M. Ni, University of Macau
Tamer Özsu, University of Waterloo

PC Co-Chairs
Wenfei Fan, University of Edinburgh
Xuemin Lin, University of New South Wales
Divesh Srivastava, AT&T Labs Research

For more details: http://conferences.cis.umac.mo/icde2019/

Data Engineering

It's FREE to join!

TCDE

tab.computer.org/tcde/

The Technical Committee on Data Engineering (TCDE) of the IEEE Computer Society is concerned with the role of data in the design, development, management and utilization of information systems.

- Data Management Systems and Modern Hardware/Software Platforms
- Data Models, Data Integration, Semantics and Data Quality
- Spatial, Temporal, Graph, Scientific, Statistical and Multimedia Databases
- Data Mining, Data Warehousing, and OLAP
- Big Data, Streams and Clouds
- Information Management, Distribution, Mobility, and the WWW
- Data Security, Privacy and Trust
- Performance, Experiments, and Analysis of Data Systems

The TCDE sponsors the International Conference on Data Engineering (ICDE). It publishes a quarterly newsletter, the Data Engineering Bulletin. If you are a member of the IEEE Computer Society, you may join the TCDE and receive copies of the Data Engineering Bulletin without cost. There are approximately 1000 members of the TCDE.

Join TCDE via Online or Fax

ONLINE: Follow the instructions on this page:

www.computer.org/portal/web/tandc/joinatc

FAX: Complete your details and fax this form to **+61-7-3365 3248**

Name _____

IEEE Member # _____

Mailing Address _____

_____

Country _____

Email _____

Phone _____

TCDE Mailing List

TCDE will occasionally email announcements, and other opportunities available for members. This mailing list will be used only for this purpose.

Membership Questions?

Xiaoyong Du
Key Laboratory of Data Engineering and Knowledge Engineering
Renmin University of China
Beijing 100872, China
duyong@ruc.edu.cn

TCDE Chair

Xiaofang Zhou
School of Information Technology and Electrical Engineering
The University of Queensland
Brisbane, QLD 4072, Australia
zxf@uq.edu.au

IEEE Computer Society
10662 Los Vaqueros Circle
Los Alamitos, CA 90720-1314

Non-profit Org.
U.S. Postage
PAID
Los Alamitos, CA
Permit 1398