Towards Interactive Data Exploration

Carsten Binnig[1,2](✉), Fuat Basık[4], Benedetto Buratti[1], Ugur Cetintemel[2], Yeounoh Chung[2], Andrew Crotty[2], Cyrus Cousins[2], Dylan Ebert[2], Philipp Eichmann[2], Alex Galakatos[2], Benjamin Hättasch[1], Amir Ilkhechi[2], Tim Kraska[2,3], Zeyuan Shang[2], Isabella Tromba[3], Arif Usta[4], Prasetya Utama[2], Eli Upfal[2], Linnan Wang[2], Nathaniel Weir[2], Robert Zeleznik[2], and Emanuel Zgraggen[2]

[1] TU Darmstadt, Darmstadt, Germany
carsten.binnig@cs.tu-darmstadt.de
[2] Brown University, Providence, USA
[3] Massachusetts Institute of Technology, Cambridge, USA
[4] Bilkent University, Ankara, Turkey

Abstract. Enabling interactive visualization over new datasets at "human speed" is key to democratizing data science and maximizing human productivity. In this work, we first argue why existing analytics infrastructures do not support interactive data exploration and outline the challenges and opportunities of building a system specifically designed for interactive data exploration. Furthermore, we present the results of building IDEA, a new type of system for interactive data exploration that is specifically designed to integrate seamlessly with existing data management landscapes and allow users to explore their data instantly without expensive data preparation costs. Finally, we discuss other important considerations for interactive data exploration systems including benchmarking, natural language interfaces, as well as interactive machine learning.

1 Introduction

Truly interactive visualization applications allow users to make data-driven decisions at "human speed," but traditional analytical DBMSs for OLAP workloads are ill-suited to serve this class of applications. Historically, DBMSs for OLAP workloads are optimized for data warehousing scenarios that can afford long data loading times (e.g., for index construction), and only have to support a fixed number of pre-defined reports. Moreover, traditional analytical DBMS implement an execution paradigm that run OLAP queries until completion before returning an exact result to the user which can take seconds or minutes on large data sets. All these reasons make traditional analytical DBMS solutions an exceptionally bad fit for interactive data exploration (IDE). At the same time, the expectation that a new system supporting interactive data exploration will replace existing data management stacks for analytics is, simply, unrealistic. Instead, a system designed specifically for interactive data exploration must integrate and work seamlessly with existing data infrastructures (e.g., data warehouses, distributed file systems, analytics platforms).

https://doi.org/10.1007/978-3-030-24124-7_11

We thus argue that we need to rethink the design of the data management stack to better support interactive data exploration scenarios. To illustrate the needs for a new backend, we illustrate an example workflow as shown in Fig. 1. In this example, a user wants to determine which features (shown as boxes on the left) in the US census dataset [19] affect whether an individual earns a salary of more than $50k annually. To answer this question, the user first drags out the `sex` attribute to the canvas (Step A) to view the distribution of males and females. The user then drags out the `salary` attribute, links these two visualizations, and selects the female bar to view the filtered salary distribution for females only (Step B). A duplicate of the `salary` visualization connected with a negated link (dotted line) allows a comparison of the relative salaries of males and females (Step C). After some analysis, the user decides to check whether an individual's `education` level coupled with `sex` has an impact on `salary`. Finally, linking `education` to each of the `salary` visualizations and selecting only individuals with a PhD (a rare subpopulation) creates a complex workflow comparing the respective salaries of highly educated males and females (Step D). From this analysis, the data seem to suggest that highly educated females earn less money annually than their male counterparts. To further explore this finding, the user might continue the analysis by testing the impact of additional attributes, applying statistical techniques (e.g., a t-test) to validate the finding, or performing various ML tasks (e.g., classification, clustering) to test other hypotheses. A more complete demonstration of this scenario in Vizdom is available at https://vimeo.com/139165014.

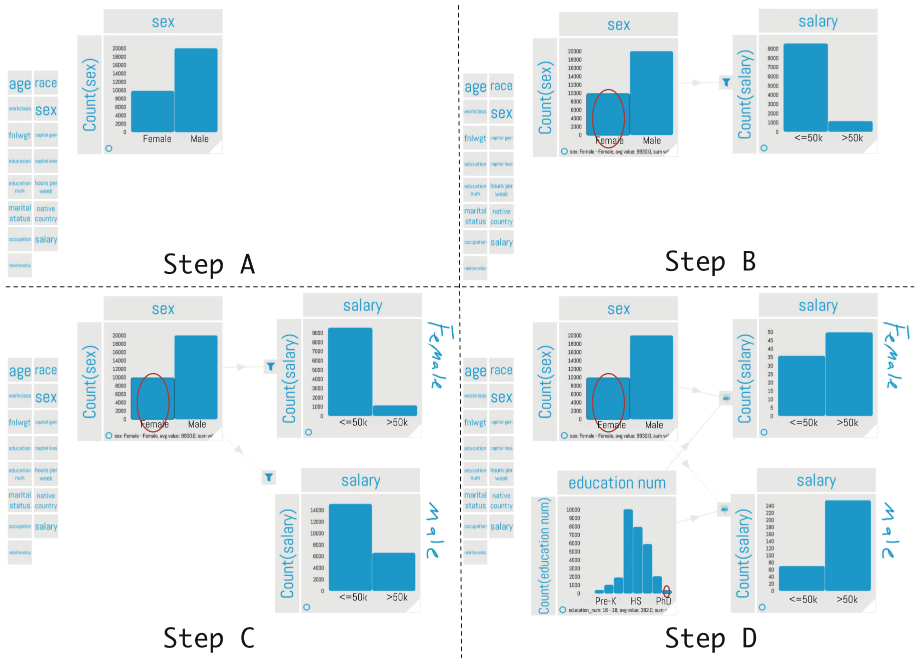

Fig. 1. Example workflow to analyze salary distributions

We therefore suggest dropping the assumptions of traditional DBMSs for OLAP workloads and propose a new breed of systems that supports (1) immediate exploration of new datasets without the need for expensive data preparation, (2) a progressive query execution model that supports interactive query response times and refines results over time, and (3) more "conversational" query interfaces that allow domain experts to incrementally explore all facets of a new data set in order to better support query processing for visual data exploration tools such as Vizdom [8].

One of the most important aspects is that throughout the data exploration process, the database backend system must be able to consistently provide response times low enough to guarantee fluid user interactions in the frontend to all queries. In fact, a recent study [20] shows that even small delays (more than 500ms) significantly decrease a user's activity level, dataset coverage, and insight discovery rate. No existing techniques, though, can guarantee interactive latencies while also addressing all of our previously stated goals. For example, existing data warehouses require a full copy of the data and suffer from long loading times, which contradicts our goal of being able to start exploring a new dataset immediately without expensive data preparation (e.g., indexing, compression). Furthermore, many existing data warehouse indexing techniques suffer from the "curse of dimensionality" and do not scale well beyond a handful of attributes [5]. Restricting the number of attribute combinations is also not an option, since the core idea of data exploration is to look for new, unexplored relationships in the data. Finally, dynamic data reorganization techniques (e.g., cracking [17]) do not solve the high-dimensionality problem and require sorting the data based on user access patterns, thereby violating the key randomness assumption of many online algorithms.

In order to return early results for long-running queries over large datasets and refine results progressively, online aggregation techniques [16,23] provide approximate answers with an estimated error, and several newer analytics frameworks (e.g., Spark [29], Flink [2]) now support online aggregation using a streaming execution model. We therefore believe that online aggregation is a good starting point since these techniques will allow the system to quickly provide initial results that are refined over time as more data is scanned. However, while online aggregation techniques work well for approximating results to queries on common subpopulations in the data, they start to break down when applied to increasingly rare subpopulations (e.g., when the user in the example selected individuals with a PhD), since scanning the base data in a random order might not provide enough of these instances to provide an accurate estimate. This problem is quite common in many interactive data exploration use cases, where rare events often contain the most interesting insights (e.g., the habits of the few highly valued customers). Although disproportionate stratified sampling can help in these cases by overrepresenting rare data items, these samples typically need to be fully constructed before data exploration even begins [1,6], contradicting our goal of enabling immediate exploration of new datasets. More importantly, though, most of these systems make the strong assumption that the entire workload is known a priori in order to create appropriate samples, whereas our goal is to allow users to explore data in new and potentially unanticipated directions.

This paper gives an overview of our keynote given at BIRTE 2017[1] and assembles results from different previously published papers [3,4,9,10,12,13,15,32]. In summary, in this paper we make the case for a new bread of systems for interactive data exploration and present the results of our own implementation called IDEA (Interactive Data Exploration Accelerator), which allows users to connect to existing data sources and immediately begin the process of interactive data exploration. The outline of the paper is organized as follows:

– We first outline the overall challenges and opportunities associated when building a new system for interactive data exploration (Sect. 2).
– We then describe the design and unique contributions of our system called IDEA (Sect. 3).
– We discuss other important considerations for interactive data exploration including benchmarking, natural language interfaces, as well as interactive machine learning and outline our contributions in those directions as well (Sect. 4).
– We finally conclude in Sect. 5.

2 Challenges and Opportunities

Designing a system for interactive data exploration with a human-in-the-loop frontend requires solving a set of very unique research challenges while also opening the door to several interesting opportunities. In this section, we first outline some of the requirements and challenges, followed by an overview of some of the unique opportunities to address them.

2.1 Challenges

Interactive data exploration has a very unique set of requirements (e.g., response time guarantees), many of which are pushing the boundaries of what is feasible today.

Interactive Latencies: By far, the most important challenge in supporting interactive data exploration is to display a result within the latency requirement. As [20] showed, even small delays of more than 500 ms can significantly impact the data exploration process and the number of insights a user makes. Therefore, a new system for IDE need to maintain certain response time guarantees in order to provide a fluid user experience. Moreover, we believe that a system should be able to refine the query answer progressively. This allows users to get a more accurate answer while visually inspecting the query results.

[1] http://db.cs.pitt.edu/birte2017/keynote.html.

Conversational Queries: Different from classical OLAP workloads, users want to explore all different factets of a data set instead of browsing a fixed set of reports. This is very different from whart existing analytical databases assume since they expect that the workload is known a priori to create the "right" indexes/samples, whereas the goal of data exploration is to explore and visualize the data in new ways. Moreover, indexes and data cubes suffer from the curse of dimensionality, since memory required is exponential with the number of attributes, making it almost impossible to build an index over all attributes or without knowing the data exploration path ahead of time.

Rare Data Items: Data exploration often involves examining the tails of a distribution to view the relatively rare data items. For example, real world datasets are rarely perfectly clean and often contain errors (which are typically rare) that can still have a profound effect on the overall results. Similarly, valid outliers and the tails of the distribution are often of particular interest to users when exploring data (e.g., the few billionaires in a dataset, the super users, the top-k customers, the day with the highest traffic). Unfortunately, for rare events and the tail of the distribution, sampling techniques do not work well since they often miss rare items or require a priori knowledge of the workload, a challenge when designing an system for IDE.

Connect and Explore: Ideally, the user should be able to connect to a dataset and immediately start exploring it. However, this requirement implies that there is no time for data preparation and the system has to build all internal storage structures such as indexes on the fly. Another implication of the *connect and explore* paradigm is that the system has to stream over larger datasets (from the sources) and may not be able to hold the entire dataset in memory (or even on disk). As outlined in the introduction, online aggregation methods are a good fit to overcome this challenge, since they provide an immediate estimate (with error bars) over the incoming stream. However, online aggregation techniques assume that the data is random, which might be false since some data sources (e.g., data warehouses) often sort the data on some attribute. This can result in a biased estimate of the result and invalid error bars. Similarly, no good estimates are possible if the source returns the data in some chronological order and if there is some (unknown) correlation between time and the value of interest (e.g., the sales are increasing over time).

Quantifying Risk: An interactive data exploration system with a visual interface allows users to explore hundreds of hypotheses in a very short amount of time. Yet, with every hypothesis test (either in the form of an explicit statistical test or through a more informal visualization), the chance of finding something by chance increases. Additionally, the visual interface can make it easier to overlook other challenges, (e.g., "imbalance of labels" for a classifier) which can lead to incorrect conclusions. Therefore, quantifying the risk is extremely important for an interactive data exploration system.

2.2 Opportunities

Although there are several challenges to address, there are many unique opportunities, since data exploration involves close interactions between analysts and the system. Many of these challenges have not yet been explored within the data management community.

Think Time: Although the user expects subsecond response times from the system, the system's expectation from the user is different; there might be several seconds (or sometimes even minutes) between user interactions. During this time, the system not only has the chance to improve the current answers on the screen, but also prepare for any future operations. For instance, in our running example, the user might have already dragged out the `sex` and `salary` attribute, but not yet linked them together. Given that both attributes are on the screen, the system might begin creating an index for both attributes. Should the user decide to link the two visualizations and use one as a filter, the index is already created to support this operation.

Interaction Times: Similar to think time, the system can also leverage the user interaction time to provide faster and more accurate answers. For example, it takes several hundred valuable milliseconds to drag an attribute on the interactive whiteboard or to link two visualizations together. In contrast to the previous think time, user interactions are much shorter but usually provide more information to the system about the intent of the user.

Incremental Query Building: In contrast to one-shot DBMS queries, data exploration is an iterative, session-driven process where a user repeatedly modifies a workflow after examining the results until finally arriving at some desired insight. For example, think of the session shown in Fig. 1 where the user first filteredn salary by gender and then added a filter on education. This session-driven discovery paradigm provides a lot of potential to reuse results between each interaction and modification.

Data Source Capabilities: Traditional analytics systems like Spark and streaming systems like Streambase assume that they connect to a "dumb" data source. However, many data sources are far from "dumb". For instance, commonly the data source is a data warehouse with existing indexes, materialized views, and many other advanced capabilities. While these capabilities do not directly fulfill the needs for interactive data exploration, they can still be used to reducing load and network traffic between the data warehouse and the accelerator. Furthermore, there has been work on leveraging indexes [22] to retrieve random samples from a DBMS. These techniques, together with the possibility to push down user-defined functions (UDFs) to randomize data, provide a feasible solution to the previously mentioned bias problem.

Human Perception: One of the most interesting opportunities stems from the fact that all results are visualized. Therefore, often precise answers are not needed and approximations suffice. Furthermore, the human eye has limitations and humans are particularly bad at understanding the impact of error bars [11]. The system can exploit both of these properties to provide faster response times (i.e., only compute what is perceived by the user).

Modern Hardware: Finally, there are several modern hardware trends that can significantly improve the amount of work that can be done in less than 500 ms. While there has been already a lot of work in leveraging GPUs for data exploration [21], most of the existing solutions focus on single machine setups and ignore the potential of small high-performance clusters. Small high performance clusters can help to significantly increase the amount of available main memory (1–2 TB of main memory is not uncommon with 8 machine cluster), which is crucial for interactive speeds, while avoiding the problems of fault-tolerance and stragglers that come with large cloud deployments. At the same time, fast network interconnects with RDMA capabilities are not only more affordable for smaller clusters, but also offer unique opportunities to decrease latencies. However, taking full advantage of the network requires carefully redesigning the storage layer of the system in order to enable remote direct memory access [4].

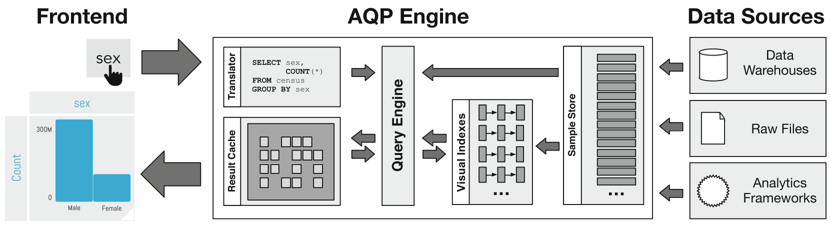

Fig. 2. The IDEA system architecture overview

3 The IDEA System

The IDEA system is the first system built specifically to enable users to visually explore large datasets through "conversational" interactions. Our prototype addresses many of the previously mentioned challenges (Sect. 2), applying novel progressive sampling, indexing, and query optimization techniques in order to provide interactive response times. In this section, we first provide an overview of our proposed architecture, followed by highlights of research insights and contributions.

3.1 Architecture

The architecture of IDEA is shown in Fig. 2. The Vizdom frontend provides a visual data exploration environment specifically designed for pen and touch interfaces, such as the recently announced Microsoft Surface Hub. A demo video of Vizdom can be found here [7]. Currently, Vizdom connects to IDEA using a standard REST interface, which in turn connects to the data sources using the appropriate protocols (e.g., ODBC). These data sources can include anything from legacy data warehouses to raw files to advanced analytics platforms (e.g., Spark [30], Hadoop [26]). As shown in Fig. 2, Vizdom connects to IDEA, which acts as an intelligent cache for those data sources and executes user queries interactively using a novel AQP engine.

IDEA's AQP engine is the core of IDEA and divides the memory into three parts: the *Result Cache*, the *Sample Store*, and space for *Indexes*. When triggered by an initial user interaction, IDEA translates it into a query and begins ingesting required data from the various data sources, speculatively performing operations and caching the results in the *Result Cache* to support possible future interactions. At the same time, IDEA also caches all incoming data in the *Sample Store* using a compressed row format. When the available memory for the *Sample Store* is depleted, IDEA starts to update the cache using a reservoir sampling strategy to eventually create a representative sample over the whole dataset. Furthermore, IDEA might decide to split up the reservoir sample into several stratified subsamples to overrepresent the tails of the distribution, or to create specialized *Indexes* on the fly to better support visual workloads. All these decisions are constantly optimized based on both past and current user interactions. For example, if the user drags a new attribute onto the canvas, the system might allocate more resources to the dragged attribute and preparation for potential follow-up queries. At the same time, IDEA constantly streams increasingly precise results to the frontend as the computation progresses over the data, along with indications about both the completeness and current error estimates.

3.2 Research Findings and Contributions

In this section, we highlight a few selected research findings and contributions of IDEA.

Progressive AQP Engine: IDEA's engine is neither a classical DBMS execution engine nor a streaming engine, instead has an entirely unique semantics. Unlike DBMSs, queries are not one-shot operations that return exact results; rather, data exploration workflows are constructed incrementally, requiring fast response times and progressive results that refine over time. At the same time, streaming engines traditionally deploy predefined queries over infinite data streams, whereas IDEA is meant to enable free-form exploration of data sampled from a deterministic system (e.g., a finite data source).

Fundamentally, IDEA acts as an intelligent, in-memory caching layer that sits in front of the much slower data sources, managing both progressive results and the samples used to compute them. Oftentimes, IDEA has the opportunity to offload pre-filtering and pre-aggregation operations to an underlying data source (e.g., perform a predicate pushdown to a DBMS), or even transform the base data by executing a custom UDF in an analytics framework. Finally, in contrast to traditional DBMSs and streaming engines, users compose queries incrementally, therefore resulting in simultaneous visualizations of many component results with varying degrees of error. Maintaining different partial results rather than a single, exact answer imposes a completely new set of challenges for both expressing and optimizing these types of queries. Currently, our IDEA prototype uses a preliminary interaction algebra to define a user's visual queries [10].

Probabilistic Query Formulation: While developing the AQP enfine of IDEA, we observed that many visualizations rely on the observed frequencies in the underlying data, or estimates of the probability of observing certain data items. For example, a bar chart over a nominal attribute is simply a visualization of the relative frequencies of the possible attribute values (i.e., a probability mass function), and a histogram of a continuous attribute visually approximates the attribute's distribution (i.e., a probability density function). Although seemingly trivial, this observation prompted us to reconsider online aggregation as a series of probability expressions.

This novel probability formulation actually permits a wide range of interesting optimizations including taking advantage of the Bayes' theorem to maximize the reuse of results. Our current implementation of IDEA therefore manages a cache of results that stores previously computed frequencies and error estimates for reuse in future queries [15].

Visual Indexes: Similar to the algebra and optimizer, we also found that traditional indexes are not optimal for interactive data exploration tasks. Most importantly, existing techniques either sort the data (e.g., database cracking) or do not naturally support summary visualizations. As previously mentioned, sorting can destroy data randomness and, consequently, the ability to provide good estimates. Similarly, indexes generally index every tuple without considering any properties of the frontend (e.g., human perception limitations, visualization characteristics). This approach often results in very large indexes, especially with increasingly large samples or highly dimensional data.

For example, some visualizations (e.g., histograms) require the system to scan all leaf pages in a traditional B-tree, since this index is designed for single range requests rather than providing visual data summaries. We therefore developed VisTrees [13], a new dynamic index structure that can efficiently provide approximate results specifically to answer visualization requests. The core idea is that the nodes within the index are "visually-balanced" to better serve visual user interactions and then compressed based on perception limitations. Furthermore, these indexes are built on the fly during the think-time of users to avoid heavy upfront cost which would violate out connect-and-explore paradigm.

Sample Store: As previously mentioned, IDEA caches as much data as possible from the underlying data sources in order to provide faster approximate results, since most data sources are significantly slower. For example, the memory bandwidth of modern hardware ranges from 40–50 GB/s per socket [4], whereas we recently measured that PostgreSQL and a commercial DBMS can only export 40–120 MB/s, even with a warm cache holding all data in memory. Although DBMS export rates may improve in the future, IDEA's cache will still remain crucial for providing approximate answers to visual queries and supporting more complex analytics tasks (e.g., ML algorithms).

If the cached data exceeds the available memory, IDEA needs to carefully evict stored tuples while retaining the most important data items in memory. For example, caching strategies like LRU do not necessarily maintain a representative sample. Therefore, IDEA uses reservoir sampling instead to evict tuples while preserving randomness. Furthermore, IDEA also needs to maintain a set of disproportionate stratified samples that overrepresent uncommon data items in order to support operations over rare subpopulations. The necessity to maintain different types of potentially overlapping samples poses many interesting research challenges. For example, deciding when and what to overrepresent is a very interesting problem, and IDEA uses a cost model to make this decision as described in [15].

Inconsistencies: Interactive response times often require computing approximate answers in parallel, which can lead to inconsistencies in concurrent views (e.g., the combined `salary` bars shown in Fig. 1(B) may not sum to the total number of females). Similarly, an outlier that appears in one result visualization may not yet be reflected in another, causing the user to draw a potentially incorrect conclusion.

Although initially assuming that inconsistencies would pose an important challenge for IDEA, we found that this problem only arises in a few corner cases, and we did not observe any consistency issues during various user studies [10,15,31]. In particular, IDEA's result reuse and sampling techniques work together to mitigate many potential consistency problems, and any noticeable differences tend to disappear before the user can even recognize them.

4 Other Considerations

In addition to the core challenges that we address in IDEA to support a backend, there are many other considerations when building a novel data management system for interactive data exploration.

4.1 Benchmarking IDE Systems

Existing benchmarks for analytical database systems such as TPC-H [28] and TPC-DS [27] are designed for static reporting scenarios. However, those benchmarks are not suitable for evaluating new backends for interactive data exploration because of different reasons. For instance, the main metric of these benchmarks is the performance of running individual SQL queries to the end, thereby not supporting more recent systems which return approximate results such as IDEA [10], approXimateDB/XDB [18], or SnappyData [24]. More importantly, workloads of traditional analytical benchmarks do not meet the complexity of actual data exploration workflows where queries are built and refined incrementally.

We have therefore started to work a novel benchmark called *IDEBench* [12] that can be used to evaluate the performance of IDE systems under realistic conditions in a standardized, automated, and re-producible way. An initial version of the benchmark and results of running the benchmark on several data analytics backends for interactive data exploration is available[2].

4.2 Natural Language Interfaces

While visual exploration tools have recently gained significant attention, Natural Language Interfaces to Databases (NLIDB) appeared as a high-promise alternative as it enables users to pose complex ad-hoc questions in a concise and convenient manner. For example, imagine that a medical doctor starts her new job at a hospital and wants to find out about the age distribution of patients with the longest stays in the hospital. This question typically requires the doctor—when using a standard database interface directly—to write a complex nested SQL query. Even with a visual exploration tool such as Tableau [25] or Vizdom [8], a query like this is far from being trivial since it requires the user to execute multiple query steps and interactions. Alternatively, with an exploration tool that provides a natural language interface, the query would be as simple as stating "What is the age distribution of patients with the longest stays in the hospital?". However, understanding natural language questions and translating them accurately to SQL is a complicated task, and thus NLIDBs have not yet made their way into commercial products.

We therefore developed *DBPal*, a relational database exploration tool that provides an easy-to-use natural language (NL) interface aimed at improving the transparency of the underlying database schema and enhancing the expressiveness and flexibility of human-data interaction through natural language. Different from existing approaches, our system leverages deep models to provide a more robust query translation. Our notion of model robustness is defined as the effectiveness of the translation model to map linguistically varying utterances to finite pre-defined relational database operations. Take, for example, a SQL expression *SELECT * FROM patients WHERE diagnosis='flu'*. There are numerous corresponding natural language utterances for this query, such as *"show all patients*

[2] https://idebench.github.io/.

with diagnosis of flu" or simply *"get flu patients"*. We aim to build a translating system that is invariant towards these linguistic alterations, no matter how complex or convoluted. The video at https://vimeo.com/user78987383/dbpal shows a recording of a representative user session in our system.

4.3 Interactive Model Curation

Extracting actionable insights from data has been left to highly trained individuals who have a background in machine learning. For example, it is common practice for corporations to employ teams of data scientists that assist stakeholders in building models to find qualitative, data-driven insights to inform possible business decisions. Having such a high-entry bar to data analysis however presents several challenges. For one, it presents a bottleneck. While research is trying to understand and promote visualization and data literacy and educational institutions are ramping up their data science curricula there is still a shortage of skilled data scientists. And second, and more importantly, restricting data analysis to those with a computational and machine learning background creates an inequality. Small business owners without those skills or research domains where computational background might not be as prevalent are at a disadvantage as they can not capitalize on the power of data.

We believe that there is an opportunity for tool builders to create systems for people who are domain experts but neither ML experts. We are therefore working on a new system for Quality-aware Interactive Curation of Models, called *QuIC-M* [3]. Through *QuIC-M* domain experts can build these pipelines automatically from high level tasks specification and at a fast pace without the need to involve a data scientist and without sacrificing quality. Making sense of data is exploratory by nature, and demands rapid iterations and all but the simplest analysis tasks, require humans-in-the-loop to effectively steer the process. *QuIC-M* exposes a simple model building interface allowing domain experts to seamlessly interleave data exploration with curation of machine learning pipelines. However, empowering novice users to directly analyze data also comes with drawbacks. It exposes them to "the pitfalls that scientists are trained to avoid" [14]. We discussed and described such "risk" factors and QuIC-M's user interface in related works [9,32].

5 Conclusion

In this paper, we presented the case for a new bread of data management systems which seek to maximize human productivity by allowing users to rapidly gain insights from new large datasets. We outlined the research challenges and opportunities when building such a new system and discussed the insights we gained from building our system called IDEA. Finally, we discussed other important considerations in the context of building interactive data exploration systems including benchmarking, natural language interfaces, as well as interactive machine learning.

References

1. Agarwal, S., et al.: BlinkDB: queries with bounded errors and bounded response times on very large data. In: EuroSys, pp. 29–42 (2013)
2. Apache Flink. http://flink.apache.org/
3. Binnig, C., et al.: Towards interactive curation & automatic tuning of ML pipelines. In: 1st Inaugural Conference on Systems ML (SysML) (2018)
4. Binnig, C., et al.: The end of slow networks: it's time for a redesign. In: VLDB, pp. 528–539 (2016)
5. Böhm, C., Berchtold, S., Kriegel, H., Michel, U.: Multidimensional index structures in relational databases. J. Intell. Inf. Syst. **15**, 51–70 (2000)
6. Chaudhuri, S., Das, G., Narasayya, V.R.: Optimized stratified sampling for approximate query processing. TODS **32**, 9 (2007)
7. Crotty, A., et al.: Vizdom Demo Video. https://vimeo.com/139165014
8. Crotty, A., et al.: Vizdom: interactive analytics through pen and touch. In: VLDB, pp. 2024–2035 (2015)
9. Crotty, A., Galakatos, A., Zgraggen, E., Binnig, C., Kraska, T.: Vizdom: interactive analytics through pen and touch. Proc. VLDB Endow. **8**(12), 2024–2027 (2015)
10. Crotty, A., Galakatos, A., Zgraggen, E., Binnig, C., Kraska, T.: The case for interactive data exploration accelerators (IDEAs). In: HILDA@SIGMOD, p. 11. ACM (2016)
11. Cumming, G., Finch, S.: Inference by eye: confidence intervals and how to read pictures of data. Am. Psychol. **60**, 170–180 (2005)
12. Eichmann, P., Zgraggen, E., Zhao, Z., Binnig, C., Kraska, T.: Towards a benchmark for interactive data exploration. IEEE Data Eng. Bull. **39**(4), 50–61 (2016)
13. El-Hindi, M., Zhao, Z., Binnig, C., Kraska, T.: VisTrees: fast indexes for interactive data exploration. In: HILDA (2016)
14. Fisher, D., DeLine, R., Czerwinski, M., Drucker, S.: Interactions with big data analytics. Interactions **19**(3), 50–59 (2012)
15. Galakatos, A., Crotty, A., Zgraggen, E., Binnig, C., Kraska, T.: Revisiting reuse for approximate query processing. PVLDB **10**(10), 1142–1153 (2017)
16. Hellerstein, J.M., Haas, P.J., Wang, H.J.: Online aggregation. In: SIGMOD, pp. 171–182 (1997)
17. Idreos, S., Kersten, M.L., Manegold, S.: Database cracking. In: CIDR, pp. 68–78 (2007)
18. Li, F., Wu, B., Yi, K., Zhao, Z.: Wander join: online aggregation via random walks. In: ACM SIGMOD, pp. 615–629. ACM (2016)
19. Lichman, M.: UCI Machine Learning Repository (2013)
20. Liu, Z., Heer, J.: The effects of interactive latency on exploratory visual analysis. TVCG **20**, 2122–2131 (2014)
21. Liu, Z., Jiang, B., Heer, J.: imMens: real-time visual querying of big data. In: EuroVis, pp. 421–430 (2013)
22. Olken, F., Rotem, D.: Random sampling from relational databases. In: VLDB, pp. 160–169 (1986)
23. Pansare, N., Borkar, V.R., Jermaine, C., Condie, T.: Online aggregation for large MapReduce jobs. In: VLDB, pp. 1135–1145 (2011)
24. Snappy data. https://www.snappydata.io/. Accessed 02 Nov 2017
25. Tableau. http://www.tableau.com. Accessed 02 Nov 2017
26. The Apache Software Foundation. Hadoop. http://hadoop.apache.org
27. TPC-DS (2016). http://www.tpc.org/tpcds/. Accessed 02 Nov 2017

28. TPC-H (2016). http://www.tpc.org/tpch/. Accessed 02 Nov 2017
29. Zaharia, M., Das, T., Li, H., Hunter, T., Shenker, S., Stoica, I.: Discretized streams: fault-tolerant streaming computation at scale. In: SOSP, pp. 423–438 (2013)
30. Zaharia, M., et al.: Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. In: NSDI, pp. 15–28 (2012)
31. Zgraggen, E., Galakatos, A., Crotty, A., Fekete, J., Kraska, T.: How progressive visualizations affect exploratory analysis. IEEE Trans. Vis. Comput. Graph. **23**(8), 1977–1987 (2017)
32. Zhao, Z., De Stefani, L., Zgraggen, E., Binnig, C., Upfal, E., Kraska, T.: Controlling false discoveries during interactive data exploration. In: Proceedings of the 2017 ACM International Conference on Management of Data, pp. 527–540. ACM (2017)