Data Vocalization with CiceroDB

Immanuel Trummer
Cornell University
Ithaca, NY, USA
itrummer@cornell.edu

ABSTRACT

Data vocalization is the process of summarizing data via voice output. We present CiceroDB, a novel database system, designed from the ground up for vocal output of query results. It is targeted at scenarios in which visual output is either impossible or undesirable. CiceroDB exploits the particularities of vocal output to reduce data processing overheads. For instance, it uses sampling to converge to high-level voice descriptions and overlaps voice output with background processing tasks. In this paper, we describe preliminary results and ongoing research efforts.

1. INTRODUCTION

The database community has almost exclusively focused on visual output when it comes to data analysis. Visual output is however not always an option, as demonstrated by the following example.

EXAMPLE 1.1. *Imagine a blind data scientist who wants to analyze mid-career salary in America based on a US Census data set. Visual output is generally not accessible to such users. Hence, a system that allows for instance the following voice interaction would be useful. User: so how does the mid-career salary depend on the start salary? System: The average mid-career salary is 80 K. Values increase by about 20% for a start salary of at least 50 K. User: ok, and how does it depend on the region? System: The mid-career salary increases by about 5% in the North-East and in California.*

Visually impaired users are not the only ones who can benefit from voice-based data analysis. The communication between user and computer is generally shifting more and more towards speech-based interfaces. Devices and services such as Google Home, Amazon Alexa, or Apple's Siri are primarily designed for vocal interaction. While average users might still opt for visual interfaces for extensive data analysis, they might use the latter to quickly satisfy their curiosity with regards to specific issues (e.g., while watching a TV documentary on climate change: *Hey Alexa, how did the average temperature develop in different parts of California over the past 10 years?*).

This article is published under a Creative Commons Attribution License (http://creativecommons.org/licenses/by/3.0/), which permits distribution and reproduction in any medium as well allowing derivative works, provided that you attribute the original work to the author(s) and CIDR 2019.
9th Biennial Conference on Innovative Data Systems Research (CIDR '19) January 13-16, 2019 , Asilomar, California, USA.

Table 1: Visualization versus vocalization.

Criterion	Visualization	Vocalization
Delivery	One-Shot	Gradual
Control	User	System
Persistency	Durable	Fleeting

We present CiceroDB, a research prototype targeted at voice-based data analysis. Prior work has focused on answering queries with small result sets via voice output [45]. Our goal is to support the analysis of large data sets instead. Our research focuses on two questions: how to summarize large query results concisely? And how to generate those descriptions efficiently?

In CiceroDB, we evaluate strategies for generating voice answers to input queries in different scenarios and for different data types. We focus on holistic vocalization methods that combine query evaluation and voice output generation. We thereby exploit the particularities of vocalization to reduce processing overheads. For instance, instead of generating full query results, we evaluate queries partially, targeting result properties that matter for high-level voice output. Also, exploiting the sequential nature of voice output, we overlap background processing with voice output to reduce latency.

In this paper, we describe ongoing research and future research plans, as well as preliminary results. We first discuss data vocalization and its particularities in Section 2. We show how those particularities influence the design of CiceroDB. In Section 3, we formalize the problem model supported by our system. We give a high-level overview of the architecture of CiceroDB in Section 4 and present first experimental results in Section 5. We discuss ongoing research in Section 6, followed by a comparison to prior work in Section 7.

2. DESIGN CONSIDERATIONS

We point out particularities that distinguish voice output from visual output in Section 2.1. In Section 2.2, we discuss implications for the design of CiceroDB.

2.1 Particularities of Vocalization

We compare vocal to visual output. Our observations apply to all forms of visual output (e.g., plots or written text).

Table 1 summarizes the following points. Visual output generally offers more control to users. Users choose themselves which part of a plot to study. They can skim written text to quickly identify relevant parts. When encountering difficult passages, they can adapt their reading speed or reread important passages. None of that is easily possible with vocal output. Here, the system controls which information is transmitted and the pace of delivery.

Visual output is typically "durable". This means it remains on display until the user chooses to switch to the next plot or text. This gives users ample time to study output parts repeatedly. Voice output, on the other hand, is fleeting. Each output part is only revealed for a short instant. Users can only work efficiently with voice output if they are able to commit at least parts of it to memory.

Finally, visual output is typically revealed in a one-shot fashion. Voice output, on the other hand, is revealed gradually. Users have no information on the text beyond the current sentence. In particular, they cannot tell whether the following sentences have already been chosen by the system.

2.2 Design Implications

The aforementioned particularities in voice output lead to specific challenges and opportunities.

Voice output is fleeting. Listeners must remember it to work effectively with it. Capacity of short-term memory is however very limited [49]. Also, in contrast to visual output, we cannot rely on users to efficiently "prune" out irrelevant information via skimming. Each additional piece of information increases the duration of voice output (and hence the burden on the listener). Both properties of voice output place tight constraints on the amount of information we can transmit. This implies the following design constraint.

Implication 2.1. *Voice output needs to focus on high-level tendencies and the system must carefully select which information to transmit.*

It is clear that we cannot talk about single tuples anymore when vocalizing a large query result. Voice output needs to describe data at a higher level of abstraction. We might not even be able to output all significant tendencies in the data. Hence, we must carefully select which information to transmit. Those insights motivate us to formalize vocalization as an optimization problem, maximizing the amount of information transmitted under constraints on speech length (see Section 3 for more details).

We are unable to transmit query results at a high degree of detail via voice output. This leads to the following insight:

Implication 2.2. *Generating complete query results with high precision is wasteful.*

We can reduce query processing overheads when taking into account the particularities of voice output. Our goal is to avoid generating result parts that do not influence the high-level voice description. CiceroDB therefore takes a holistic approach to vocalization, combining data processing and voice output generation. This distinguishes our approach from prior work on vocalization that uses complete query results as input [76].

Finally, we established in the last subsections that the delivery modes of visual and vocal output differ: visual output appears typically at once while vocal output is delivered gradually. This leads to the following opportunity:

Implication 2.3. *We can overlap incremental background processing with voice output.*

CiceroDB generally generates voice output incrementally. As discussed in more detail in Section 4, we can "pipeline" generation and output of speech fragments. By continuing background processing while voice output is already playing, we gain significant amounts of additional processing time. To the best of our knowledge, CiceroDB is the first system to exploit this possibility (which is specific to vocalization).

3. PROBLEM MODEL

CiceroDB summarizes query results via voice output. We formalize voice output generation as an optimization problem in the following.

As input, we are given a query q on the current database (CiceroDB allows users to specify that query directly or to translate voice input to queries via a keyword-based mechanism). We assume that users analyze data in an interactive session. By H, we denote the history of queries and result descriptions generated in the current session.

We are given a search space $\mathcal{S}(q, H)$ of candidate voice descriptions. The search space depends on the current query q but also on the session history H (e.g., we can refer back to previous results to shorten the current voice description). The function that generates the search space depends on the scenario. A forthcoming publication [75] describes a first search space targeted at exploratory analysis of large relational data sets. In the future, we plan to allow users to specify customized search spaces to extend CiceroDB to new scenarios.

We denote the length of a speech $s \in \mathcal{S}(q, H)$ via $|s|$. CiceroDB allows to set a threshold on the length of generated speeches. We denote this threshold by L in the following. The goal of voice output is to summarize $\mathcal{R}(q)$, the result of executing q on the database. Note that CiceroDB does in general not generate this result in its entirety.

Our goal is to "educate" users about the query result as much as possible, given the constraints on output length. Beyond the information explicitly given, users may infer new information or make default assumptions. For instance, users tend to make uniformity assumptions in the absence of further information. This principle has been formalized as the "maximum entropy principle" and is often used in the context of visual OLAP [47, 64]. We capture such effects by a user model $\mathcal{U}(s, H)$ modeling the belief on the query result of a typical user after listening to speech s, given query history H. We can shorten voice descriptions by omitting pieces of information that users can infer themselves.

Our goal is to approach the user's belief as much as possible to the true query result. We designate by $\mathcal{D}(\mathcal{U}(s, H), \mathcal{R}(q))$ the distance between user belief and actual result. We are now ready to formalize the problem that CiceroDB focuses on.

Definition 3.1 (Holistic Vocalization). *Given query q with result $\mathcal{R}(q)$, a query history H, a speech length threshold L, and functions $\mathcal{S}$, $\mathcal{U}$, and $\mathcal{D}$ capturing speech search space, user belief, and distance respectively, our goal is to find the speech $s^* \in \mathcal{S}(q, H)$ minimizing distance between user belief and query result:*

$$s^* = \arg \min_{s \in \mathcal{S}(q,H), |s| \leq L} \mathcal{D}(\mathcal{U}(s, H), \mathcal{R}(q))$$

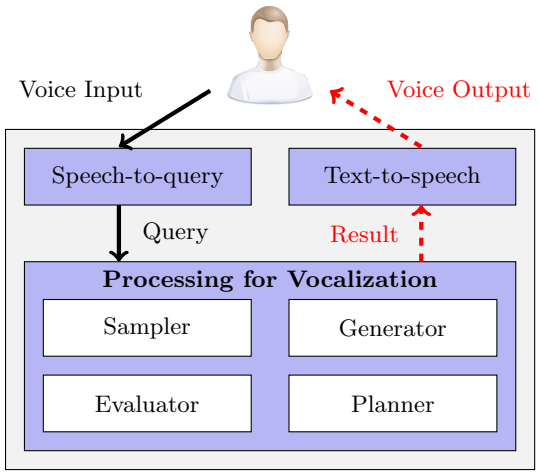

Figure 1: Overview of CiceroDB (pipelined data transmission marked up in red).

We introduced a generic problem model, not specifying the precise user model, speech search space, and distance function used. A forthcoming publication [75] introduces concrete functions for large-scale exploratory data analysis. Our goal is to enable users to easily define their own functions in the future. The term "holistic" in our problem definition refers to the fact that we combine data processing with result vocalization. This distinguishes our work from prior work starting from a given query result [76].

4. CICERODB OVERVIEW

Figure 1 shows a high-level overview of CiceroDB. Users issue voice queries. We currently support simple SQL queries with equality predicates, grouping, and aggregation. Input speech is transcribed via an existing speech-to-text service[1]. Next, we use a simple, keyword-based mechanism to translate input speech to an SQL query. Alternatively, we allow users to enter SQL queries directly via keyboard. Our research focus in CiceroDB is the summarization of query results via voice output (not the translation of voice input into SQL queries).

The SQL query forms the input to the processing engine. Our processing engine is holistic and does not separate query processing and voice output generation into two phases. This allows us to restrict query processing to result aspects that are relevant for voice output. CiceroDB avoids generating result details that cannot be transmitted via voice output anyway. Also, holistic processing allows us to exploit gradual delivery of voice output. CiceroDB pipelines generation and output of speech fragments (currently at the granularity of sentences). This means that data processing for the next sentence proceeds while the current sentence is being spoken out. Typically, this gives us several tens of seconds of additional processing time (the typical length of generated speeches) without introducing noticeable latency.

The holistic processing engine leverages a standard relational database system for data storage and initial access (we currently use Postgres [59]). During processing, we may load part of the data set into main memory to support fast sampling. The processing engine can be divided further into four sub-components: generator, sampler, evaluator, and planner. The generator generates candidate speech fragments for voice output. Admissible speech fragments are determined by the input query and previously spoken speech fragments (as voice output is produced incrementally). The sampler retrieves samples from the query result, either by issuing queries to the underlying database system or by accessing the in-memory buffer. The evaluator updates quality estimates of specific speech fragments based on newly retrieved samples. Associating speech fragments with precise quality values would require us to generate the entire query result (since speech quality is based on how well speech approximates the true result). Instead, we calculate confidence bounds on speech quality based on result samples. The planner controls the query evaluation process. In particular, it picks speech fragments for which to refine quality estimates and decides which samples to retrieve. The current planner is based on Monte-Carlo Tree Search [34]. It selects speech fragments to assess based on a metric balancing exploration (selecting speech fragments about which little is known) and exploitation (refining estimates for promising speech fragments). We plan to evaluate alternative approaches in the future.

[1]https://cloud.google.com/speech-to-text/

5. FIRST RESULTS

We are currently implementing multiple vocalization methods in CiceroDB. In the following, we present extracts of a vocalization method targeted at exploratory OLAP-style analysis of large relational data sets. We call this method OLAP-Vocalizer in the following. More details on the approach as well as more results can be found in an upcoming publication at SIGMOD 2019 [75].

The OLAP-Vocalizer summarizes results of OLAP-style queries: aggregate values, broken down by multiple dimensions. Speech output summarizes high-level tendencies in the result. Each speech starts with a summary of typical aggregation values in the current result (e.g., *"The average mid-career salary is 50K"*). Each of the following sentences targets a subset of aggregates, defined by values in a subset of dimensions. For those aggregates, we express how their average differs from the general average (e.g., *"The mid-career salary increases by 20% for computer scientists from the North-East."*). The problem model is one instance of the generic model described in Section 3. We select speeches based on a distance metric, measuring how well a speech approximates a query result, and a user belief model. The latter simulates the belief of users about the query result after listening to a speech. It takes into account information that is not explicitly given but can be inferred by users (e.g., having a mid-career salary above average in one region implies a salary below average in the remaining regions). Speech generation follows the high-level process described in Section 4: we evaluate speech quality via sampling and generate the speech sentence by sentence, overlapping speaking time with background processing.

We present in the following two key results from our experimental evaluation. Those results show the benefit of holistic processing for voice output (instead of separating query processing and voice output generation into two phases). They also show that users prefer the voice descriptions generated by CiceroDB over prior work.

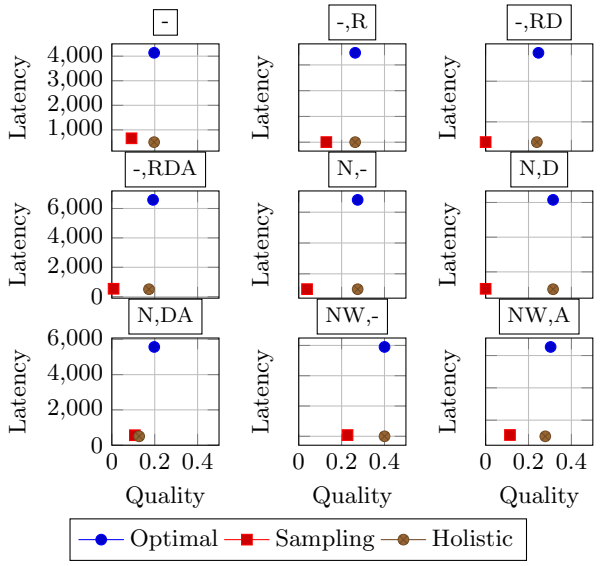

Figure 2: Performance of vocalization variants.

Figure 2 compares the holistic OLAP-Vocalizer ("Holistic" in Figure 2) against two baselines. We executed queries on a 600 MB data set about flight cancellations in 2015[2]. Queries calculate average flight cancellation probability for data subsets. Plot titles describe queries concisely, specifying first the query predicates (N for a restriction to the North-East region, W for a restriction to flights in Winter) then the dimensions by which results are broken down (R, D, and A representing a breakdown by airport region, flight date, and airline respectively). Experiments were executed on a MacBook Air computer with a 2.2 GHz Intel Core i7 processor and 8 GB of RAM.

We compare according to two metrics: the latency in milliseconds (i.e., how long the system needs before speech starts after the input query is received) and speech quality (i.e., quality is higher, the closer the speech approximates the actual query result). We compare against a baseline ("Optimal" in Figure 2) that processes the input query in its entirety and calculates quality estimates for each possible speech, thereby identifying the optimal speech. We also compare against another baseline ("Sampling" in Figure 2) that samples the query result for 500 ms (the threshold for interactive data analysis [44, 50, 66]). Next, it selects the optimal speech based on this limited sample (i.e., we do not overlap processing with voice output).

Clearly, holistic processing realizes an interesting tradeoff between latency and speech quality. The generated speeches are in most cases identical to the optimal speech according to our model. In the remaining cases, quality estimates are very close. At the same time, latency perceived by users is very low. Both non-holistic baselines perform badly according to one of the two comparison metrics. Generating the optimal speech leads to prohibitive latency for interactive analysis. Generating speeches based on a small sample leads to very low quality.

We also performed a user study with 20 crowd workers (recruited on Amazon Mechanical Turk[3]), whose task was

[2]https://www.kaggle.com/usdot/flight-delays
[3]https://www.mturk.com/

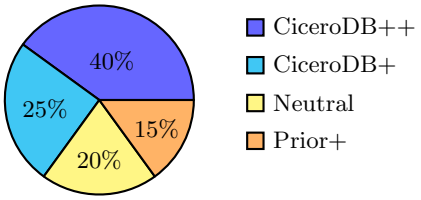

Figure 3: Percentage of crowd workers expressing strong (++) or some (+) preference for CiceroDB or prior methods when analyzing a large data set.

to analyze the aforementioned data set about flight delays via an online version of CiceroDB. Workers had the option to switch between two vocalization methods, the one introduced in CiceroDB and a previously proposed method [76]. The latter method is not targeted at large query results and generated speeches target single values, as opposed to high-level tendencies. According to Figure 3, most users prefer vocalization methods that are tailored to large data sets. Altogether, those results validate our focus on holistic vocalization and on high-level voice descriptions.

6. ONGOING WORK

In the following, we discuss several ongoing research efforts that expand the capabilities of CiceroDB.

6.1 "Tweening" for Vocalization

Data tweening has recently been proposed in the context of data visualization [30]. Data tweening is targeted at interactive data analysis where users issue multiple, related queries in a sequence. The goal is to illustrate the transition between consecutive query results via an animation. We are currently working on approaches that transfer this principle to data vocalization. Our goal is to describe differences between consecutive query results via voice output. Hence, instead of vocalizing each result separately, we focus on the "delta". This approach is motivated by the fact that users tend to issue many similar queries during an analysis session [30]. In data vocalization, few result details can be transmitted without overwhelming listeners. Hence, it is critical to restrict output to the most important facts and to avoid redundancies. Data tweening for vocalization is first of all a mechanism to avoid redundancies. We avoid redundant output of result properties that remain the same across multiple queries. Second, the same motivation as for visual tweening applies [30]: we hope to enhance the users comprehension of query transformations and data.

6.2 Exact Data Vocalization

Currently, CiceroDB uses sampling to quickly generate approximate results. Approximate results may not be acceptable in all situations (e.g., consider the example of a surgeon, using a voice interface to retrieve critical data during a procedure [45]).

We could use standard query processing first and vocalize later. This option neglects however the sequential nature of voice output: it is unnecessary to have all results available once voice output starts. Instead, it is beneficial to overlap processing and output. We are working on query planning methods that take those particularities into account. Our planning goal is to minimize latency by query processing

until voice output can start. Additionally, our goal is to avoid any interruptions after voice output has begun.

To that purpose, we consider sequences of processing steps during planning. Each processing step generates parts of the results that are needed for voice output. Each processing step is realized by a traditional query plan. Processing and voice output need to be aligned for optimal performance. Ideally, we generate all required information for the first output sentence with negligible execution overheads. Then, the time required to read each output sentence is ideally sufficient to generate enough information for the next sentence at least. The plan generating all required results with minimal overheads is not necessarily the best plan for voice output. If the plan generates no useful results until processing is terminated, its execution time turns into latency, perceived by the user. If, on the other side, total execution time is large but results become available incrementally in a suitable order for voice output, perceived latency can decrease. In our ongoing efforts, we are developing cost models and optimization methods, that allow to optimally align (exact) query processing and voice output.

7. RELATED WORK

The database community has produced a large body of work on how to optimally present relational data to users. So far, the focus has been nearly exclusively on visual data representation [27, 32, 70, 77, 81]. The lack of appropriate methods for voice output of relational data has been hinted at in prior work [72], as well as its unfortunate implications from the perspective of inclusion. First voice-based query interfaces have appeared quite recently [8, 45]. In contrast to CiceroDB, they are not targeted at the analysis of large data sets (they assume relatively small query results). Also, they do not support holistic vocalization (i.e., they do not use specialized processing strategies for voice answering). Prior work on vocalization is either limited to small query results [76] or specific to time series data [74].

Our work is complementary to prior work on data visualization. In particular, CiceroDB relates to prior systems that use processing methods, tailored for generating visualizations [27, 29, 32]. Similar in intent to CiceroDB, the goal is to reduce processing overheads compared to generic processing strategies. Here, we apply a similar reasoning to the problem of generating data vocalizations.

Prior work on rendering relational data as natural language text [71] differs from our work in terms of focus (text for visual output instead of audio output), scope (output of static data instead of interleaved query evaluation and output generation), and method (no multi-objective optimization). The various particularities that come into play when generating text for voice output (as opposed to text for visual consumption) have been discussed extensively in prior work [4, 60].

Most query results need to be summarized for voice output. The goal in intensional query answering [1, 6, 9, 11, 12, 47, 53, 54, 56, 67, 68] is to summarize extensional results by a succinct, intensional representation. Our work differs again by its focus on voice output, scope, and methods. Research on natural language query interfaces [2, 42, 43, 45, 63] has mainly focused on challenges in parsing natural language queries (or, alternatively, on rendering SQL queries as natural language text [35, 36]). In CiceroDB, we prioritize the complementary research question of how to optimally summarize results. Reading out one table row after another is only possible for very small data sets [45].

Our work relates to a lesser degree to research outside of the database community which does not target relational data. Prior work on transforming data into voice output focuses on text data [3, 21], HTML code [25, 26, 46, 58], or math formulas [17, 60, 73]. Sonification [23] typically designates approaches that render data via non-voice audio (an early definition excludes voice output explicitly [37] while a recent definition [22] is less restrictive). For instance, numerical data can be translated into notes where pitch correlates with the numerical value [5, 31, 61]. The sweet spot of most of those approaches are numerical data, in particular data which could be represented as a plot. This covers only a small part of relational data sets. Clearly, voice output is more amenable for mainstream applications.

Natural language generation methods are categorized as text-to- text or data-to-text methods [19]. Text-to-text methods [13, 65, 69] use unstructured information as input and relate less to our research. Research on data-to-text natural language generation [20, 24, 28, 38, 48, 51, 62] is often targeted at the generation of written reports as opposed to voice output. Corresponding approaches neglect issues related to query processing and typically use highly domain-specific rules or templates to summarize large data sets (e.g., for weather-forecasts [20] or neonatal care [24]). The generation of spoken text from data was examined in the context of spoken dialogue systems. This line of research typically exploits user preferences [7, 52, 79, 80] or multiple dialogue steps [10, 14, 15, 57] to narrow down a set of options (e.g., alternative flights) for the user. The proposed work is based on different assumptions (e.g., data points do not necessarily represent alternative choices) and scope (e.g., dialogue planning is out of scope). Our research is complementary to prior work focusing on the question of how users perceive alternative summaries of structured data [39, 40, 41] (instead of how to generate such summaries efficiently).

Finally, our research connects to prior work in audiology and psychology. Research in those areas has contributed empirical and theoretical results linking various aspects of speech to intelligibility and comprehension. For instance, sentence length [78], word choice [16], and complexity of sentence structure [55] were all shown to have impact on comprehension. Of particular relevance to this work are results that apply specifically to intelligibility of automatically generated speech [18, 33, 78]. We exploit results from those areas for the design of our search space for speech output.

8. CONCLUSION

CiceroDB is a research prototype for voice-based analysis of large data sets. It answers queries via concise voice descriptions. Its query processing engine is tailored to the particularities of voice output and exploits them to reduce processing overheads. Preliminary results support the design decisions made in CiceroDB. In ongoing projects, we are expanding its capabilities along multiple dimensions.

9. REFERENCES

[1] A. C. Acar and A. Motro. Intensional encapsulations of database subsets via genetic programming. In *DEXA*, pages 365–374, 2005.

[2] I. Androutsopoulos, G. D. Ritchie, and P. Thanisch. Natural language interfaces to databases - an

introduction. *Journal of Natural Language Engineering*, 1(1):29–81, 1995.

[3] B. Arons. SpeechSkimmer: a system for interactively skimming recorded speech. *ACM Transactions on Computer-Human Interaction*, 4(1):3–38, 1997.

[4] B. M. Arons. *Interactively skimming recorded speech*. PhD thesis, 1994.

[5] J. Batterman and B. Walker. Auditory graphs need error bars: validating error-to-sound mappings and scalings. In *ICAD*, pages 315–318, 2013.

[6] F. Benamara. Generating intensional answers in intelligent question answering systems. In *International Natural Language Generation Conference*, pages 11–20, 2004.

[7] G. Carenini and J. D. Moore. An empirical study of the influence of argument conciseness on argument effectiveness. In *IJCAI*, pages 150–157, 2001.

[8] D. Chandarana, V. Shah, A. Kumar, and L. Saul. SpeakQL: towards speech-driven multi-modal querying. In *HILDA*, pages 1–6, 2017.

[9] W. W. Chu, R.-C. Lee, and Q. Chen. Using type inference and induced rules to provide intensional answers. In *ICDE*, pages 396–403, 1991.

[10] G. Chung. Developing a flexible spoken dialog system using simulation. In *Annual Meeting on Association for Computational Linguistics*, pages 63–70, 2004.

[11] P. Cimiano, H. Hartfiel, and S. Rudolph. Intensional question answering using ILP: what does an answer mean? In *Natural Language and Information Systems*, pages 151–162, 2008.

[12] P. Cimiano, S. Rudolph, and H. Hartfiel. Computing intensional answers to questions: an inductive logic programming approach. *Data & Knowledge Engineering*, 69(3):261–278, 2010.

[13] J. Clarke and M. Lapata. Discourse constraints for document compression. *Computational Linguistics*, 36(3):411–441, 2010.

[14] V. Demberg and J. D. Moore. Information presentation in spoken dialogue systems. In *EACL*, pages 65–72, 2006.

[15] V. Demberg, A. Winterboer, and J. D. Moore. A strategy for information presentation in spoken dialog systems. *Computational Linguistics*, 37(3):480–539, 2011.

[16] F. Ferreira, J. M. Henderson, M. D. Anes, P. A. Weeks, and D. K. Mcfarlane. Effects of lexical frequency and syntactic complexity in spoken-language comprehension: evidence from the auditory moving-window technique. *Journal of Experimental Psychology*, 22(2):324–335, 1993.

[17] H. Ferreira and D. Freitas. Enhancing the accessibility of mathematics for blind people: The audiomath project. In *International Conference on Computers for Handicapped Persons*, pages 678–685, 2004.

[18] A. L. Francis and H. C. Nusbaum. The effect of lexical complexity on intelligibility. *International Journal on Speech Technology*, 3(1):15–25, 1999.

[19] A. Gatt and E. Krahmer. Survey of the state of the art in natural language generation: core tasks, applications and evaluation. *arXiv preprint arXiv:1703.09902*, pages 1–111, 2017.

[20] E. Goldberg, N. Driedger, and R. Kittredge. Using natural-language processing to produce weather forecasts. *IEEE Expert-Intelligent Systems and their Applications*, 9(2):45–53, 1994.

[21] J. Guerreiro and D. Gonçalves. Text-to-speeches: Evaluating the perception of concurrent speech by blind people. In *ACM SIGACCESS Conference on Computers and Accessibility*, pages 169–176, 2014.

[22] T. Hermann. Taxonomy and definitions for sonification and auditory display. In *International Conference on Auditory Display*, pages 1–8, 2008.

[23] T. Hermann, A. Hunt, and J. G. Neuhoff. *The Sonification Handbook*. 2011.

[24] J. Hunter, Y. Freer, A. Gatt, E. Reiter, S. Sripada, and C. Sykes. Automatic generation of natural language nursing shift summaries in neonatal intensive care: BT-Nurse. *Artificial Intelligence in Medicine*, 56(3):157–172, 2012.

[25] F. James. Presenting HTML structure in audio: user satisfaction with audio hypertext. In *ICAD*, pages 97–103, 1996.

[26] F. James. Lessons from developing audio HTML interface. In *ACM Conference on Assistive Technologies*, pages 27–34, 1998.

[27] U. Jugel, Z. Jerzak, and G. Hackenbroich. M4 : A visualization-oriented time series data aggregation. *PVLDB*, 7(10):797–808, 2014.

[28] J. Kalita. Automatically generating natural language reports. *International Journal of Man-Machine Studies*, 30(4):399–423, 1989.

[29] N. Kamat and A. Nandi. InfiniViz: Interactive Visual Exploration using Progressive Bin Refinement. *arXiv preprint arXiv:1710.01854*, 2017.

[30] M. Khan, L. Xu, A. Nandi, and J. Hellerstein. Data tweening: incremental visualization of data transforms. *PVLDB*, 10(6):661–672, 2017.

[31] J. Kildal and S. a. Brewster. Providing a size-independent overview of non-visual tables. In *ICAD*, pages 8–15, 2006.

[32] A. Kim, E. Blais, A. Parameswaran, P. Indyk, S. Madden, and R. Rubinfeld. Rapid sampling for visualizations with ordering guarantees. *PVLDB*, 8(5):521–532, 2015.

[33] E. A. M. Klabbers and R. P. G. Collier. On the performance of speech output in a practical setting. *IPO Annual Progress Report*, 33:121–128, 1995.

[34] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. In *European Conf. on Machine Learning*, pages 282–293, 2006.

[35] A. Kokkalis, P. Vagenas, A. Zervakis, A. Simitsis, G. Koutrika, and Y. Ioannidis. Logos: A system for translating queries into narratives. In *SIGMOD*, pages 673–676, 2012.

[36] G. Koutrika, A. Simitsis, and Y. E. Ioannidis. Explaining structured queries in natural language. In *ICDE*, pages 333–344, 2010.

[37] G. Kramer, B. Walker, P. Coordinator, T. Bonebright, P. Cook, J. Flowers, N. Miner, J. Neuhoff, R. Bargar, S. Barrass, J. Berger, G. Evreinov, W. T. Fitch, M. Gröhn, S. Handel, H. Kaper, H. Levkowitz, S. Lodha, B. Shinn-cunningham, M. Simoni, and

S. Tipei. *Sonification report: status of the field and research agenda*. 1999.

[38] K. Kukich. Design of a knowledge-based report generator. In *Annual Meeting on Association for Computational Linguistics*, pages 145–150, 1983.

[39] B. Langner. *Data-driven natural language generation: making machines talk like humans using natural corpora*. PhD thesis, 2010.

[40] B. Langner and A. W. Black. uGloss: a framework for improving spoken language generation understandability. In *INTERSPEECH*, pages 2893–2896, 2007.

[41] B. Langner, R. Kumar, A. Chan, L. Gu, and A. W. Black. Generating time-constrained audio presentations of structured information. In *INTERSPEECH*, pages 2450–2453, 2006.

[42] F. Li and H. Jagadish. NaLIR: an interactive natural language interface for querying relational databases. *SIGMOD*, pages 709–712, 2014.

[43] F. Li and H. Jagadish. Understanding natural language queries over relational databases. *SIGMOD Record*, 45(1):6–13, 2016.

[44] Z. Liu and J. Heer. The effects of interactive latency on exploratory visual analysis. *IEEE Transactions on Visualization & Computer Graphics*, 20(12):2122–2131, 2014.

[45] G. Lyons, V. Tran, C. Binnig, U. Cetintemel, and T. Kraska. Making the case for Query-by-Voice with EchoQuery. In *SIGMOD*, pages 2129–2132, 2016.

[46] J. Mahmud, Y. Borodin, I. V. Ramakrishnan, and D. Das. Combating information overload in non-visual web access using context. In *IUI*, pages 341–344, 2007.

[47] P. Marcel, P. J. Jaurès, and S. Rizzi. Towards intensional answers to OLAP queries for analytical sessions. In *DOLAP*, pages 49–56, 2012.

[48] K. McKeown, J. Robin, and K. Kukich. Generating concise natural language summaries. *Information Processing and Management*, 31(5):703–733, 1995.

[49] G. A. Miller. The magical number 7, plus or minus 2 - some limits on our capacity for processing information. *Psychological Review*, 63(2):81–97, 1956.

[50] R. B. Miller. Response time in man-computer conversational transactions. In *AFIPS*, pages 267–277, 1968.

[51] M. Molina, A. Stent, and E. Parodi. Generating automated news to explain the meaning of sensor data. In *Intelligent Data Analysis*, pages 282–293, 2011.

[52] J. Moore, M. E. Foster, O. Lemon, and M. White. Generating tailored, comparitive descriptions in spoken dialogue. In *AAAI*, pages 917–922, 2004.

[53] A. Motro. Using integrity constraints to provide intensional answers to relational queries. In *VLDB*, pages 237–246, 1989.

[54] A. Motro. Intensional answers to database queries. *Transactions on Knowledge and Data Engineering*, 6(3):444–454, 1994.

[55] E. Murphy. *The effect of working memory and syntactic complexity on sentence comprehension*. PhD thesis, 2013.

[56] E. K. Park and S.-C. Yoon. An approach to intensional query answering at multiple abstraction levels using data mining approaches. In *HICSS*, pages 9–17, 1999.

[57] J. Polifroni, G. Chung, and S. Seneff. Towards the automatic generation of mixed-initiative dialogue systems from Web content. In *Eurospeech*, pages 193–196, 2003.

[58] E. Pontelli, D. Gillan, G. Gupta, and A. Karshmer. Intelligent non-visual navigation of complex HTML structures. *Universal Access in the Information Society*, 2(1):56–69, 2002.

[59] PostgreSQL Global Development Group. PostgreSQL. https://www.postgresql.org/, 2017.

[60] T. V. Raman. *Audio system for technical readings*. PhD thesis, 1998.

[61] R. Ramloll, W. Yu, and B. Riedel. Using non-speech sounds to improve access to 2D tabular numerical information for visually impaired users. In *Conference of the British HCI Group*, pages 515–529, 2001.

[62] E. Reiter, R. Robertson, and L. Oman. Types of knowledge required to personalise smoking cessation letters. In *AIMDM*, pages 389–399, 1999.

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

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

[65] B. P. Sharifi, D. I. Inouye, and J. K. Kalita. Summarization of twitter microblogs. *Computer Journal*, 57(3):378–402, 2014.

[66] B. Shneiderman. Response time and display rate in human performance with computers. *ACM Computing Surveys*, 16(3):265–285, 1984.

[67] C. Shum and R. Muntz. Implicit representation for extensional answers. In *Expert Database Systems*, pages 257–273, 1988.

[68] C.-D. Shum and R. Muntz. An information-theoretic study on aggregate responses. In *VLDB*, pages 479–490, 1988.

[69] A. Siddharthan and M. A. Angrosh. Hybrid text simplification using synchronous dependency grammars with hand-written and automatically harvested rules. In *Annual Meeting on Association for Computational Linguistics*, pages 722–731, 2014.

[70] T. Siddiqui, J. Lee, A. Kim, E. Xue, C. Wang, Y. Zou, L. Guo, C. Liu, X. Yu, K. Karahalios, and A. Parameswaran. Fast-forwarding to desired visualizations with zenvisage. In *CIDR*, 2017.

[71] A. Simitsis, Y. Alexandrakis, G. Koutrika, and Y. Ioannidis. Synthesizing structured text from logical database subsets. In *EDBT*, pages 428–439, 2008.

[72] A. Simitsis and Y. Ioannidis. DBMSs should talk back too. In *CIDR*, 2009.

[73] R. D. Stevens, D. E. Alistair, and P. A. Harling. Access to mathematics for visually disabled students though multimodal interaction. In *HCI*, pages 47–92, 1997.

[74] I. Trummer, M. Bryan, and R. Narasimha. Vocalizing large time series efficiently. *VLDB*, 11(11):1563–1575, 2018.

[75] I. Trummer, Y. Wang, and S. Mahankali. A holistic

approach for query evaluation and result vocalization in voice-based OLAP. In *SIGMOD*, pages 1–18, 2019.

[76] I. Trummer, J. Zhu, and M. Bryan. Data vocalization: optimizing voice output of relational data. *PVLDB*, 10(11):1574–1585, 2017.

[77] M. Vartak, S. Madden, A. Parameswaran, and N. Polyzotis. SeeDB: automatically generating query visualizations. *VLDB*, 7(13):1581–1584, 2014.

[78] H. Venkatagiri. Effect of sentence length and exposure on the intelligibility of synthesized speech. *Augmentative and Alternative Communication*, 10(2):96–104, 1994.

[79] M. A. Walker, S. J. Whittaker, A. Stent, P. Maloor, J. Moore, M. Johnston, and G. Vasireddy. Generation and evaluation of user tailored responses in multimodal dialogue. *Cognitive Science*, 28(5):811–840, 2004.

[80] M. White, R. A. J. Clark, and J. D. Moore. Generating tailored, comparative descriptions with contextually appropriate intonation. *Computational Linguistics*, 36(2):159–201, 2010.

[81] K. Wongsuphasawat, D. Moritz, A. Anand, J. Mackinlay, B. Howe, and J. Heer. Voyager: exploratory analysis via faceted browsing of visualization recommendations. *Transactions on Visual and Computer Graphics*, 22(1):649–658, 2015.