To appear in *International Journal of Speech Technology*.

A Multi-Modal Natural Language Interface to an Information Visualization Environment

Kenneth Cox[†], Rebecca E. Grinter[*]
Stacie L. Hibino[†], Lalita Jategaonkar Jagadeesan[†], and David Mantilla[‡]

[†]Bell Labs, Lucent Technologies
263 Shuman Boulevard
Naperville, IL 60566 USA
{kcc, hibino, lalita}@research.bell-labs.com

[*]Xerox PARC
3333 Coyote Hill Road
Palo Alto, CA 94304 USA
grinter@parc.xerox.com
(work conducted at Bell Labs)

[‡]Department of Computer Science
Harvard University
Cambridge, MA 02138
mantilla@fas.harvard.edu
(work conducted at Bell Labs)

ABSTRACT

Domain experts frequently know what questions they want to ask about a data set, but they do not necessarily know the mechanisms of using an information visualization (infoVis) system for investigating these inquiries of interest. In our work, we are researching the addition of a natural language (NL) interface for bridging this gap between NL questions and data exploration within an infoVis environment. In this paper, we present our approach to integrating an NL interface into an existing infoVis system. We illustrate the power of combining direct manipulation with NL interfaces, and we provide a real-life example of applying our approach to the analysis of a large, complex data set.

Keywords: information visualization, speech and natural language interfaces, multi-modal interfaces

1. INTRODUCTION

When analyzing a large, complex data set, domain experts usually know what questions they have, but not necessarily how to translate them into the system commands required by an information visualization (infoVis) system. Unfortunately, many infoVis systems only allow users to interact with the data by formulating complex queries (in, for example, SQL) and creating and manipulating views (e.g., such as scatterplots or network visualizations). Thus, domain experts require knowledge of database querying and various available visualizations in order to benefit from infoVis. For example, an average investor may know what questions they want to ask about their portfolio or about current market trends, but probably do not typically know (nor want to learn) how to pose SQL queries or select different types of visualizations. In this paper, we present an alternative interface to an infoVis system—one that allows users to simply interact with their data through a natural language (NL) interface.

Despite improvements in NL systems, it seems almost counterintuitive to use the spoken word to manipulate and explore data. Indeed, the introduction of speech-based interfaces caused us to make two design decisions early. First, should we *replicate* direct manipulation commands with speech? Previous studies found that this kind of one-to-one replication is not effective [8, 16], and that speech-based interfaces should exploit the inherent characteristics of natural language—characteristics such as context and dialog [11]. Second, should we *replace* direct manipulation with speech, or provide a fully multi-modal system? Again, previous research suggests that a multi-modal system offers end users with the richest experience since it provides opportunities to move among the modes according to the work being done [17].

In this paper, we present our approach to integrating a NL interface into an existing infoVis system. Our approach offers three advantages. First, our system is fully multi-modal, allowing users to move among different input modes according to their preference. These modes include a traditional point-and-click interface, NL input in textual form via a web page or GUI, and NL via speech.

Second, by introducing NL, we remove the burden from the user of translating intuitively simple questions into complex queries and view selections required by the system. Specifically, our system takes the natural language question, determines the appropriate database query, and presents the appropriate results to the user (e.g., by providing a spoken response or presenting one or more visualizations for the user to explore with respect to their question). Our system provides an interactive approach where users have a dialogue with the system as they explore their data.

Third, our system combines domain-independent and domain-specific NL components. We have created a set of domain independent speech commands that allow users to explore data by referring to details of the presentation. The advantage here is that these speech commands apply to any data set under investigation. We have augmented this approach by providing a domain-specific, NL component that allows users to ask questions about the data with no regard to the infoVis component. This domain-specific interface needs to be customized for the data under investigation, but frees the user from having to understand how to query infoVis systems. Thus, users can focus on their questions rather than the system details.

Using this approach, we have built a natural language interface to an existing infoVis system, and have used it to analyze a significantly large Lucent Technologies' corporate database.

Overview. This paper is divided into six additional sections. In the next section, we describe the methodology we used in developing this tool and the dialogs it uses, and indicate how this methodology has broader applications. Section 3 provides additional details of our implementation. Section 4 is a walk-through of a data analysis using the system. In Section 5, we present results of a pilot usability study. In Section 6, we discuss some advantages and disadvantages of the natural-language interface. In Section 7, we discuss some related work and finally in Section 8, we present our conclusions and describe some future work.

2. OUR APPROACH

The goal of this work is to support the analysis of very large data sets. Based on our experiences, we have found that novice infoVis users typically know what questions they want to ask, but often do not know how to express these questions in a form that is suitable for a given infoVis environment or other analysis tool. In addition, when such a tool is demonstrated to users, the users quickly recognize the value of the visual results and usually do not have problems interacting with these results in search of answers to their proposed questions. The key barrier to the use and adoption of infoVis environments for complex analysis is thus the number of system details that users must attend to before they can actually begin to explore their data within a visual environment. In other words, there is a gap between the user's expectations and the tool's capabilities.

In our lab, we are currently investigating several avenues for addressing this problem. For example, we designed and developed a web-based approach to interactive information visualization in context, which we refer to as *LiveDocs* [15]. We are also working on a new framework for using information visualization to support the end-to-end *process* of data analysis [6]. We refer to this framework as *InfoStill* (short for Information Distillery). In this paper, we now present another approach that we are currently investigating—the addition of a natural language interface to an information visualization system.

In this section, we describe the reasoning behind our architecture and the requirements for the system components. In our approach, we assume the availability of three modular components:

- a natural language (NL) interface equipped with automatic speech recognition (ASR) for accepting user input (e.g., NL input via a microphone or typed in via a keyboard),
- an interactive NL dialog manager for obtaining user inquiries or requests and translating them into commands for the infoVis system,
- and an interactive infoVis framework for visually presenting results—a framework that is equipped with an API for external control and manipulation.

In the implementation of our approach, we used IBM's ViaVoice for ASR, the Sisl framework [3] for NL dialog management, and the InfoStill framework [6] for presenting a visualization of results. The API to InfoStill enables external systems such as Sisl to send commands directly to InfoStill. (For more details about our system, see Section 3.)

Although we used specific software systems to validate and test our approach, our approach is not limited to these systems. That is, our approach is a general one, in that other implementations (e.g., other infoVis systems) could apply our approach to adding an NL interface to their system.

The key contribution of our approach focuses on the design and development of the NL dialog manager to an infoVis system. More specifically, we focused our efforts on identifying the types of NL dialogs that would help bridge the gap between NL inquiries posed by naïve users and the display and exploration of appropriate visualizations for investigating those inquiries. We developed a dual approach to designing these NL dialogs. Our approach divides the NL dialogs into those which are independent of the data being investigated, and those which are dependent on the data.

2.1 Data-Independent NL Dialogs

Most data-analysis tools provide a large number of commands for loading, manipulating, and presenting data. The users of the tools are usually required to learn some large subset of these commands in order to perform analysis tasks. This is a reasonable approach for a person specializing in data-analysis, but for general users may require too large an investment to be worthwhile.

The first part of our dual approach to NL dialog design seeks to make the analysis tools more accessible, by providing a simpler and more user-friendly interface to the analysis tool's commands. Because these NL dialogs depend only on the tool, and not on the data being analyzed, we refer to them as data-independent NL dialogs.

The concept of data-independence is perhaps best illustrated by an example. Consider Figure 1, which is a bar chart produced by the InfoStill system. This bar chart shows some organizational data, specifically the frequency distribution of people in a certain organization by their former department. Thus, the bar chart indicates that people currently in organization X came from a number of different departments. The largest bar is "NONE," and thus shows that there were many new hires into organization X.

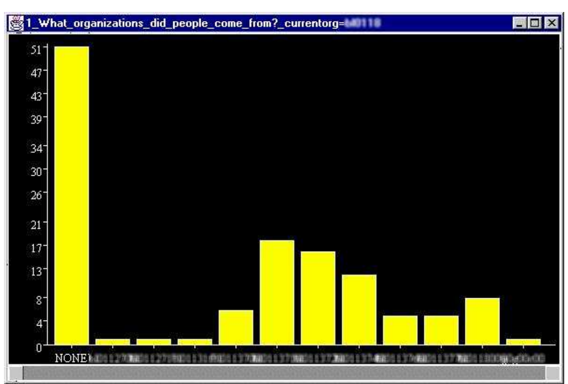


Figure 1. Bar chart view.

The InfoStill system does not know anything about the interpretation of the data it is displaying. To the system, Figure 1 is simply a display showing a summary of a data table. Any data could be shown in the bar chart, regardless of source or semantics.

The same is true of the ways in which the user can interact with a *view* such as the bar chart. In InfoStill, for example, the user can choose a particular view, select items within that view, get additional information about certain items, and perform a large number of view-specific actions, such as sorting the bars in a bar chart by height. The ability to perform these operations does not depend in any way on the data that the view is showing.

These data-independent tool commands are usually well-defined, and indeed form the bulk of the tool and its capabilities. Writing NL dialogs to access these commands is thus fairly straightforward, and often these dialogs can be based directly on the existing mouse and menu GUIs.

However, it is also possible to move beyond GUI equivalents and leverage NL features. In order to do this, additional analysis of the tool and of common applications is needed. This analysis identifies ways in which the tool and user interfaces can be used to complement one another.

In our case, for example, we worked with information visualization experts to identify common NL expressions for talking about views and interacting with them. We also identified the types of commands that are easy to speak but difficult to perform with the GUI and vice-versa. Using these results, we added NL dialogs for recognizing common NL view commands. Some of these were direct equivalents of GUI commands, while others were for operations that are cumbersome with the GUI. We were also able to avoid wasting effort on trying to provide NL equivalents for operations that are better suited to the GUI.

The example of selecting bars in a bar chart provides examples of all three cases:

- *GUI and NL equivalent:* Most GUI menu commands fell into this category, as might be expected since menu commands are often presented to the user as text strings. The menu operation “select all bars”, for example, can be spoken as easily as it can be picked from a list. Note that the GUI and NL commands are equivalent, but that the NL interface is not limited to a 1-to-1 translation of the GUI command, since the NL dialog manager can provide flexibility by allowing variations of the same command.
- *GUI superior to NL:* Many typical mouse operations fall into this category. This is particularly true for items that are simple to point to but difficult to articulate aloud (e.g., trying to select a specific bar with a long encoded label).

- *NL superior to GUI:* We identified three kinds of NL commands that may be difficult or impossible to specify with a GUI, but could be easily handled with an NL interface:

  1. *Specifying more precise commands in data selection for large data sets.* For example, the mouse can be used to choose a subset of the bars by lassoing (sketching a path that intersects the bars of interest). The user is typically interested in choosing bars by some property, e.g., “select the ten largest bars” or “select all bars that are greater than 20”. These selections are simple to say, but can be more cumbersome to perform with the mouse, especially when there are large numbers of bars, or many similarly-sized bars.
  2. *Specifying multiple commands at once, without having to wait for each command to be processed.* For example, the user might say “sort list by the first and third column,” which encapsulates two distinct commands into one sentence.
  3. *Specifying any combination of full and partial commands while the system maintains context.* Consider the above example of sorting. The user might then say “And by the fifth column.” The system maintains context, and hence can determine that the user is describing further sorting parameters. Alternatively, the user might give purely partial information such as “sort”. The system will then prompt the user for the desired sorting parameters, such as by order or by size.

Guided by the above criteria, we designed NL dialogs to support many of the GUI’s data-independent commands, as well as many data-independent commands that were not provided by the GUI but were easy to specify in NL form. The NL commands can be used *in conjunction* with the GUI commands, allowing the user to speak NL commands where appropriate or convenient, and use the GUI where suitable or preferred.

The result is a powerful NL interface to the information visualization tool, but it still falls short of the goal of answering user’s questions about their data. In order to do the latter, we need to move beyond providing access to the data-independent capabilities of the tool and develop NL dialogs for analyzing the data.

2.2 Data-Dependent NL Dialogs

The NL dialogs for mapping users’ NL questions about their data into tool commands use domain knowledge about the syntax and semantics of the data, as well as knowledge about the tool, to aid users in exploring a data set from the given domain. The aim is to allow users who are only familiar with the contents of the data and who want to learn more about trends or investigate other questions about that data, but are not familiar enough with particular analysis tools to use them for exploring the data set.

In particular, our approach uses the following steps:

- Select a data set for which to build a natural language interface.
- Work with domain experts to identify key threads of inquiry for this data set.
- For each thread of inquiry, determine what information is required to answer the question (for example, what database query parameters must be supplied).
- For each thread of inquiry, identify the sequences of tool commands that are helpful in providing relevant answers to the inquiry.
- Build dialogs to determine, from the NL input, which thread of inquiry is of interest to the user, to obtain the information required by the thread, and to issue the necessary sequences of tool commands.

We will give a brief example of this approach here; Section 4 presents a longer walk-through which illustrates some aspects of it in greater detail. Using the above approach, we built NL dialogs for analyzing a Lucent Technologies' corporate personnel database. This database, called POST, contains information about every Lucent employee. The POST database contains more than a hundred thousand entries, and is frequently updated.

One interest that researchers have with this data set involves studying how changes in the organizational structure influences the products built. This hypothesis is based on a widely known rule: Conway's Law, which says that the structure of software is the same as the structure of the organization that designed it [7]. However, while the law is well known, much less is understood about what happens as the organization evolves, and understanding that evolution requires visualizing changes in the employee database, which reflects the structure of the company.

By interviewing experts in POST and in organizational structure, we identified some typical inquiries for this data. These include questions such as[1] "What departments were the people now in department *X* in last March?" and "How did organization *X* change during 1999?" By generalizing these questions, we obtained both threads of inquiry and the information that must be obtained for each thread. For example, the first question is a thread which can be called "*peoplefrom,*" and in order to answer it we need to know the department *X* and the date.

For each thread of inquiry, we identified the class of database queries and InfoStill views that would be helpful in providing relevant answers about the POST data. To answer "*peoplefrom,*" we first need to query the database for all people currently in the given organization, then query it for the organizations that those people were in as of the given date. This

---

[1] For proprietary reasons, we cannot reveal the actual organization names or numbers.

information can then be presented using a view such as a bar chart. (The choice of a view which is most appropriate for answering a particular question is also data-dependent.)

We then implemented the NL dialogs by building an appropriate service logic for Sisl. The service logic collects information from the user through the NL interface. Its first task is to identify the user's thread of inquiry; for example, a question like "Where did the people in organization X come from?" could correspond to "*peoplefrom*," or to a related line of inquiry about the geographic location of the person. The dialog would thus prompt the user to clarify this information. Further NL dialog refines the user's statements into complete and unambiguous queries. The service logic then generates appropriate commands and sends them to InfoStill to create and display views of the data. The user can then "drill down" into more details about the data, by asking the system additional questions. Importantly, the user has access to both the NL and the GUI commands for interacting with the system.

Using this methodology, we developed NL dialogs to answer each of the above threads of inquiry (as well as others not listed here). It must again be emphasized that these NL dialogs are tailored to the particular data set, incorporating not only knowledge of the type of questions that the user might ask but also knowledge of the POST database. The result is an analysis system which allows even users with no knowledge of either POST or information visualization tools to ask NL questions about organizational structure and change, and receive answers in the form of graphical displays (i.e., visualizations). The multi-modal aspects of the system allow users to seamlessly move between the different modes (NL and GUI), and select the more natural or efficient mode for accomplishing their task.

3. SYSTEM DESCRIPTION

The overall architecture of our system is presented in Figure 2. In terms of our approach described in Section 2, we use IBM's ViaVoice for ASR, with input from a speech grammar. Because of Sisl's multi-modal interface capabilities (described further below), we can also accept NL input in textual form through a web page or GUI, as well as accepting speech for ASR through either a microphone or telephone.

We use InfoStill [6] as our information visualization system. InfoStill's internal architecture has a Task Manager (TM) which performs database queries and creates views, and a Presentation Manager (PM) which controls the views.

Finally, the NL dialog manager is implemented using Sisl [3]. The Sisl service logic obtains natural language input from the user, prompting for needed information as required, and generates commands to send to InfoStill.

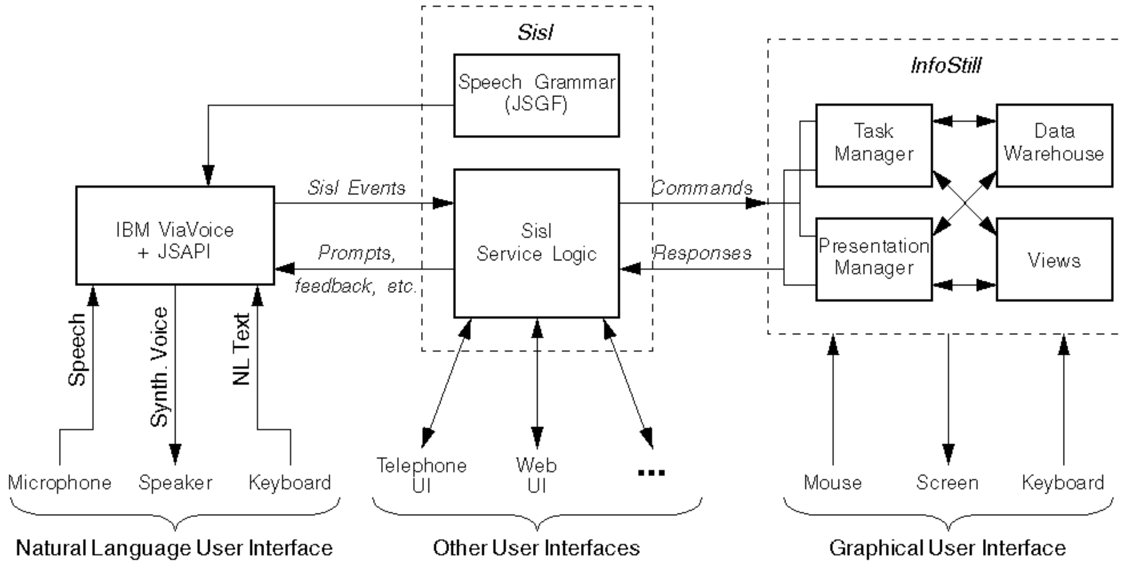


Figure 2. Overall System Architecture.

In the remainder of this section, we provide more detailed information about Sisl, InfoStill, and the integration of these two key components.

3.1 Sisl: Several Interfaces, Single Logic

Sisl is an architecture and a domain-specific language for structuring services with multiple user interfaces [3]. Duplication is a problem in this context: there is a different service logic (i.e., the code that defines the essence of the service) for every different user interface to the service. Furthermore, to support natural language style interfaces, services must allow users freedom in input by supporting different orderings of information, partial information together with early error detection, correction of information, lookahead, and reverting back to earlier points in the service. Current approaches for multi-modal services are based on finite-state machines, where every possible ordering of information must be described explicitly, and hence the resulting finite state machines are huge and impossible to maintain.

Sisl allows flexibility in inputs and allows multiple interchangeable interfaces to be added to a single consistent source of service logic and data. For example, a bank using Sisl could provide a single service logic used by all transactions, whether those transactions were accessed through an automated teller machine, bank-by-phone interface, or web-based interface. By using Sisl, several interfaces can be provided to the same centralized service logic. Thus, when a change in the service logic is required, the change only needs to be made in one place and all of the associated interfaces are automatically updated appropriately.

The service logic and user interfaces communicate using service-defined events, each carrying some information. The service logic tells the user interfaces what events it is ready to accept next. The user interfaces then prompt the user to provide the needed information, collect the information from the user, and send it to the service logic. Any user interface that performs these functions can be used in conjunction with a Sisl service logic. The Sisl service logic hence functions as a dialog manager for the interactive service.

The user interface designer need only specify, for each of the system events, a prompt and help function that generate strings to be given as prompts to the user. For NL interfaces, a set of speech grammars (the input to a speech recognition engine that permits it to efficiently and effectively recognize spoken input) is also required. Sisl automatically generates infrastructure for the user interfaces from these functions and grammars, and integrates them with the service logic.

We use the data-specific NL dialog examples of Section 2 to illustrate the use of Sisl to program NL dialogs. These NL dialogs are based on the lines of inquiry, "Where did people in organization *X* come from?" and "How has organization *X* evolved over time?" A grammar capturing these two questions is:

```
<peoplefrom> =
  ("where"
       | ("what organization"{organizations})
       | ("what location"{locations})
       )
  "did people" {who} ["in" <org> {which}]
  "come from" {peoplefrom};
<orggrowth> =
  "how has" <org> {which}
  "grown over time" {orggrowth};
```

The above syntax places non-terminals in angle brackets <>, quotes terminals expected in the NL input, and marks Sisl events with curly braces {}. Thus, if a user said "Where did people in <organization X> come from", three Sisl events would be produced: *who*, *which* (carrying the information identifying <organization X>), and *peoplefrom*.

A portion of the Sisl service logic for these inquiries is shown in Figure 3. In the initial state, the service logic can accept events corresponding to the types of inquiries, as labeled on the outgoing arcs. Given the above NL question, the *peoplefrom* transition is enabled, so Sisl changes state to the target node. At that node, it requires *who* and *which* events, in either order. As both are available, the service logic continues.

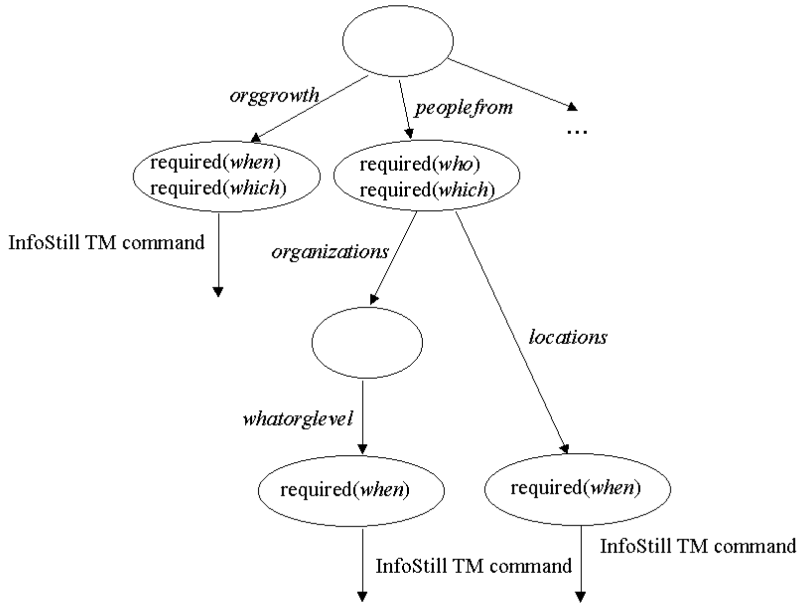


Figure 3. Sisl logic for analyzing organizational data

Sisl then finds that it does not have either an *organizations* or a *locations* event (the two allowed events at the current state). Sisl therefore informs the NL interface that it can accept either of these events. The interface then prompts the user, for example by asking "Do you mean organization or location?" This processing continues until the traversal reaches a leaf node, and an appropriate sequence of commands is sent to InfoStill. Additional details about Sisl are available in [3].

3.2 InfoStill

InfoStill, short for Information Distillery, is a framework for infoVis that aids users in the various tasks associated with performing data analyses. Previous work in infoVis has traditionally focused on only one or two stages in the analysis process, with particular emphasis on new ways of displaying data and new frameworks for exploration (e.g., [1, 4]). In practice, however, exploration is only a small fraction of an analyst's time and effort [10]. Analysts need tools to assist with all stages of the analysis process; InfoStill is an approach to providing such a tool.

InfoStill is both a workspace where users can interactively analyze data using linked views (similar to EDV [21]), and a reporting system which allows the results of analyses to be saved and examined later (similar to Crystal Reports [19] or LiveDocs [15]). InfoStill uses a linked-views paradigm, in which users can interact with a view to select some portion of the data, and this selection information is automatically propagated to all other views using that data. In other words, when a change occurs in one view, say as a result of a selection, then that selection is highlighted in all the other linked views. This lets the user look at the same set of data from many different perspectives.

For example, Figure 4 shows part of the report produced for a question about jobs within an organization. The top bar chart shows the departments within the organization; the second bar chart shows the distribution of jobs; and the table shows the names of the employees. A selection operation has been performed on the top bar chart to choose the largest department; the other views reflect this selection. We can see that the selected department has a relatively large portion of the personnel with the “Technical” job classification, as indicated by the proportion of the “Technical” bar that is shaded.

InfoStill has an API for sending commands to the system. For example, the API can be used to create a database query, create a view showing the results of the query, and perform a specific operation on a view (such as sorting a bar chart). This capability was very useful when integrating Sisl with InfoStill.

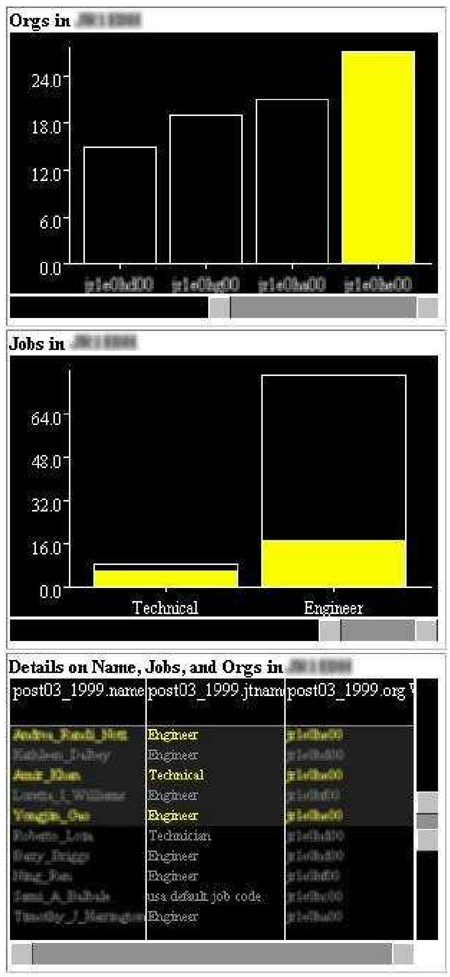


Figure 4. Portion of an InfoStill presentation, showing three linked views of organizational data.

3.3 Integrating Sisl and InfoStill

We connect Sisl to InfoStill via the InfoStill command API, thereby keeping a simple interface between the two systems. This means that we did not make InfoStill a Sisl user interface, so InfoStill does not generate Sisl events, nor does it respond to Sisl prompts for input. Instead, Sisl uses InfoStill as a system to generate and display views of data, in much the same way that a Sisl banking logic would access and use the bank's accounting system to inspect and modify account balances. This also means that when InfoStill is run with Sisl, users can interact with InfoStill through both the GUI included with InfoStill and the alternative interaction modes provided by Sisl.

4. SAMPLE SCENARIO

We now describe an actual scenario of our system for a user following lines of inquiry to Lucent's POST database of organizational data. Note that in our system, a transcript of the dialogue between the user and the system is presented in separate text window. This allows users to verify that the system has correctly interpreted their NL input (i.e., when the user provides speech NL vs. textual NL input). The transcript window also maintains the history of inquiries for the users, allowing them to easily review previous questions and feedback. If users need help, they can simply say "What can I say?" to request help from the system at any point during the dialogue. The system will then provide context-sensitive help according to the Sisl specification provided.

A Representative Dialogue

User: where did people in organization X come from?

Let us assume that the "organization X" name matches successfully against the <org> subrule. The query then matches the <peoplecomefrom> rule. In this case, the Sisl infrastructure automatically sends the following set of events to the Sisl service logic: *peoplecomefrom,* and *who*(people), and *which*(organization X). After processing these events, the Sisl logic specifies that there is a choice between organizations or locations. To disambiguate, the service prompts (using the prompt function) the user with:

Service: Do you mean organizations or locations?

User: Show me organizations.

An *organizations* event is now sent to the service logic, which prompts with:

Service: At what organizational level?

User: At the department level.

A *whatorglevel(*department*)* is now sent to the service logic, which now needs to collect information from the user to satisfy the *when* constraint. It thus prompts the user with:

Service: As of when?

User: As of now.

The Sisl service logic indicates that a complete and unambiguous query has now been specified. It then uses domain-specific knowledge to determine the kind of view that should be displayed (in this case, a bar chart), and sends a corresponding command to InfoStill. InfoStill then displays the view named View 1 in Figure 5.

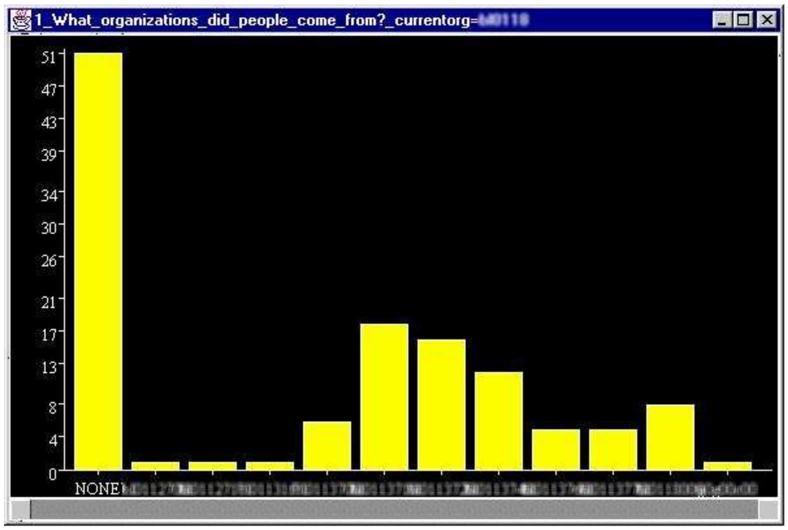


Figure 5. Bar chart for organizational inquiry; shows frequency distribution of people currently in organization X, according to their previous department (i.e., shows which departments people came from before joining organization X).

Providing Additional Information Early

The user can also specify additional information at any point in the dialogue. For example,

User: What departments did people in organization X come from?

In this case, the *whatorglevel*(departments)event is also sent to the service logic at the outset. Hence, the service then immediately jumps to asking:

Service: As of when?

In this manner, the Sisl service automatically supports a form of lookahead: users can specify information beyond what is currently needed by the service logic. If the user then specifies “as of now,” InfoStill displays the same View 1 as above.

Manipulating Views via PM Commands

The user may now interact with and manipulate these views using NL or the mouse.

User: Select view 1 and sort by count.

The Sisl service logic then sends a command to InfoStill to update View 1 as shown in Figure 6. The user could execute the same command with the mouse by using the right mouse button to access and select from a pop-up menu.

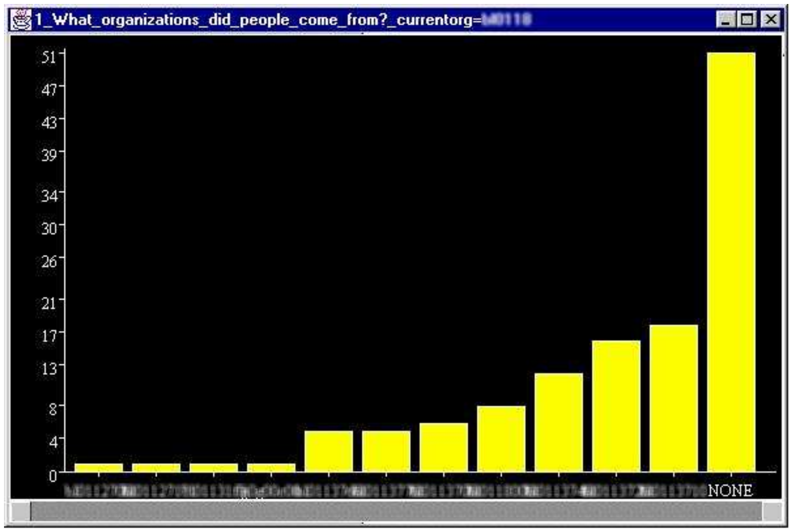
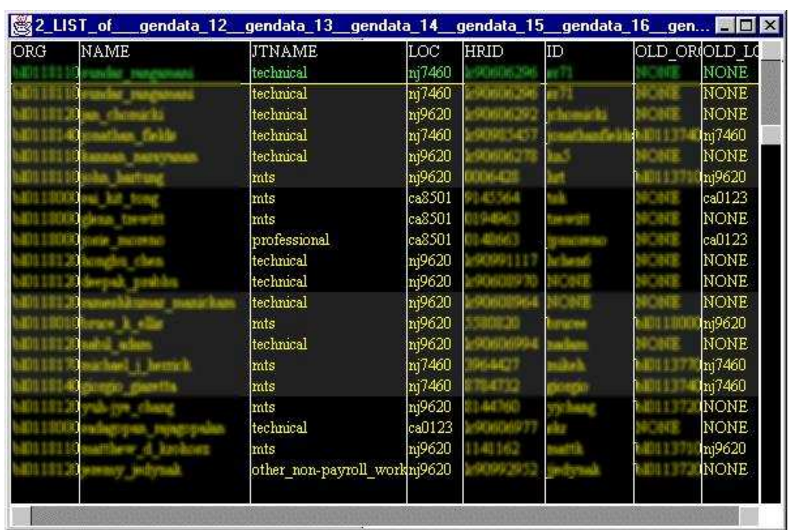


Figure 6. Sorted bar chart for organizational inquiry.

The user may then say:

User: Show me details of this data.

The Sisl service logic remembers that the selected view is View 1, and sends a command to InfoStill to display View 2, the values list shown in Figure 7. The values list is a spreadsheet-like view that provides details about individual records of the underlying data displayed in View 1. The values list in View 2 is automatically linked to View 1 so that selections made to either view (via NL or direct manipulation) are automatically reflected in the other view.



ORG	NAME	JTNAME	LOC	HRID	ID	OLD_OR(	OLD_L(
[illegible]	[illegible]	technical	nj7460	[illegible]	[illegible]	[illegible]	NONE
[illegible]	[illegible]	technical	nj7460	[illegible]	[illegible]	[illegible]	NONE
[illegible]	[illegible]	technical	nj9620	[illegible]	[illegible]	[illegible]	NONE
[illegible]	[illegible]	technical	nj7460	[illegible]	[illegible]	[illegible]	nj7460
[illegible]	[illegible]	technical	nj9620	[illegible]	[illegible]	[illegible]	NONE
[illegible]	[illegible]	mts	nj9620	[illegible]	[illegible]	[illegible]	nj9620
[illegible]	[illegible]	mts	ca8501	[illegible]	[illegible]	[illegible]	ca0123
[illegible]	[illegible]	mts	ca8501	[illegible]	[illegible]	[illegible]	NONE
[illegible]	[illegible]	professional	ca8501	[illegible]	[illegible]	[illegible]	ca0123
[illegible]	[illegible]	technical	nj9620	[illegible]	[illegible]	[illegible]	NONE
[illegible]	[illegible]	technical	nj9620	[illegible]	[illegible]	[illegible]	NONE
[illegible]	[illegible]	technical	nj9620	[illegible]	[illegible]	[illegible]	NONE
[illegible]	[illegible]	mts	nj9620	[illegible]	[illegible]	[illegible]	nj9620
[illegible]	[illegible]	technical	nj9620	[illegible]	[illegible]	[illegible]	NONE
[illegible]	[illegible]	mts	nj7460	[illegible]	[illegible]	[illegible]	nj7460
[illegible]	[illegible]	mts	nj7460	[illegible]	[illegible]	[illegible]	nj7460
[illegible]	[illegible]	mts	nj9620	[illegible]	[illegible]	[illegible]	NONE
[illegible]	[illegible]	technical	ca0123	[illegible]	[illegible]	[illegible]	NONE
[illegible]	[illegible]	mts	nj9620	[illegible]	[illegible]	[illegible]	nj9620
[illegible]	[illegible]	other_non-payroll_work	nj9620	[illegible]	[illegible]	[illegible]	NONE

Figure 7. Corresponding values list.

Another Example

As an another example, suppose the user says

User: How has [organization x] grown over time?

The query matches the <orggrowth > rule. In this case, the Sisl infrastructure automatically sends the following set of events to the Sisl service logic: *orggrowth* and *which*(organization X). After processing these events, the Sisl logic specifies that the when constraint still needs to be satisfied. It thus prompts the user with:

Service: As of when?

User: As of now.

The Sisl service now determines that a complete query has been specified and sends appropriate command(s) along to InfoStill. Two of the generated views appear in Figure 8 and Figure 9.

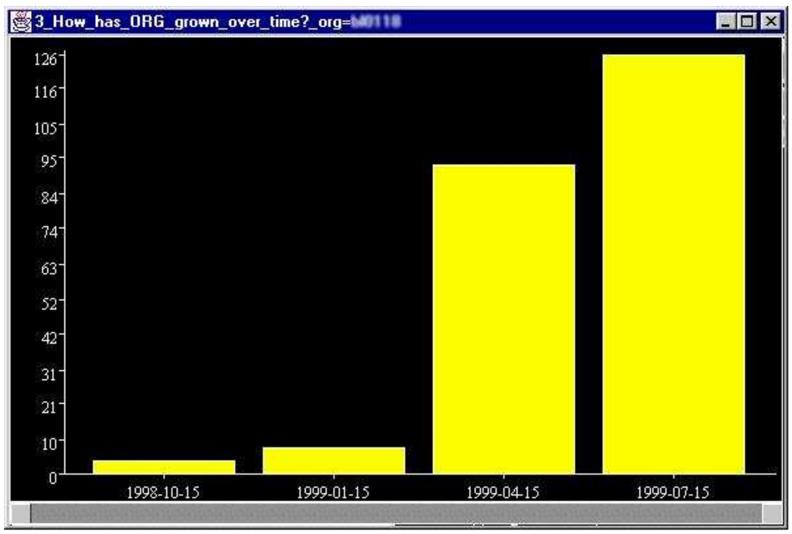


Figure 8. Bar chart for organization over time inquiry.

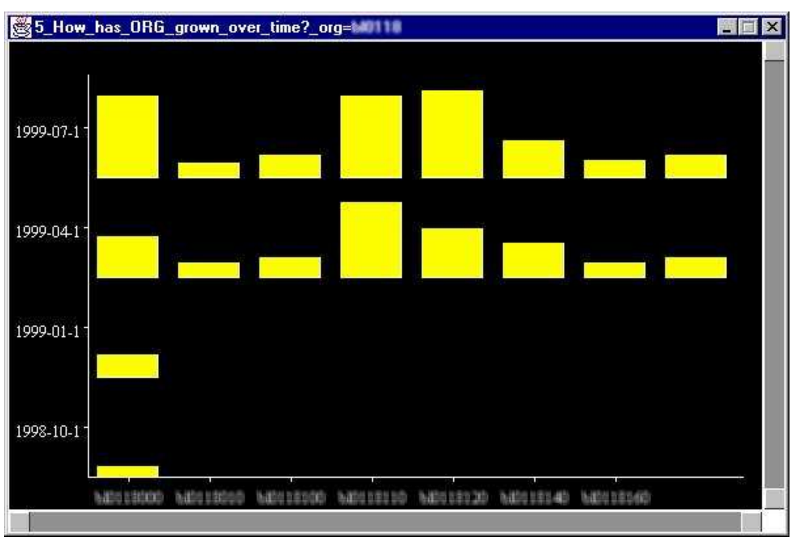


Figure 9. Grid chart for organization over time inquiry.

The user might now be interested in seeing the actual names of the employees and so might say

User: Show me details for view 3.

This values list is displayed below in Figure 10.

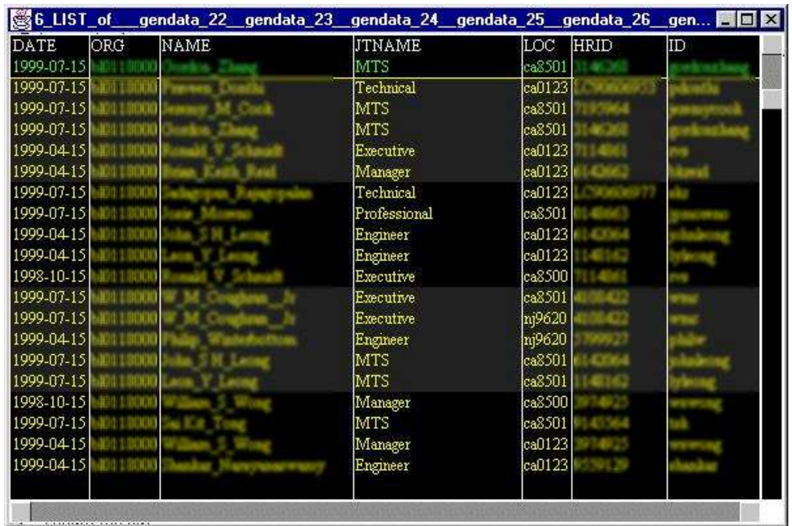


Figure 10. Corresponding values list.

Similarly to Views 1 and 2, the values list in View 6 (Figure 10) is automatically linked to the bar chart and grid chart in Figures 8 and 9. Thus, changing the selection in one view automatically changes the selection in the others. For example, if the user says:

User: Select the largest bar in View 3.

In addition to selecting the rightmost bar of View 3, the other two linked views are updated as shown in Figure 11. Again, users can manipulate the views via NL or direct manipulation, according to their preference.

Finally, the user may ask a simple question:

User: "How many people are in organization X?"

In this case the system will reply with (via speech)

System: There are 125 people in organization X.

Some grammars are dynamically generated from the dataset. For example, the references to individual views (e.g. "select view 3") are generated automatically, as are the corresponding grammars. Similarly, the <org> subrule, which specifies the organization numbers, could very easily be automatically generated from the appropriate field in the POST database. In addition to speaking the above NL queries, the user may also enter these queries in NL text form through a web page.

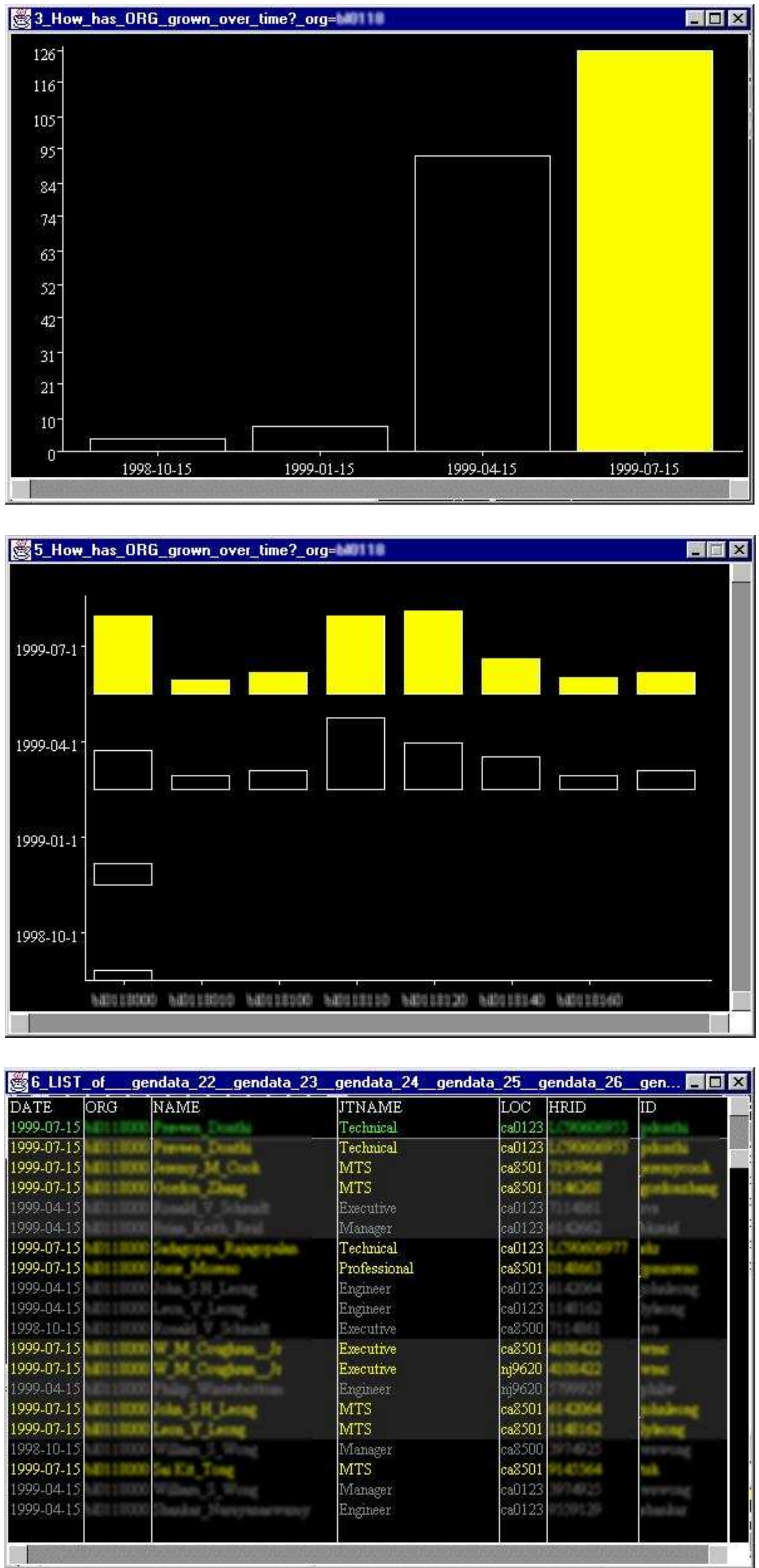


Figure 11. Updated linked views after selection.

5. PILOT STUDY

5.1 Overview of the Study

We recently conducted a small pilot study to determine 1) what types of questions general users are interested in posing about organizational data and 2) the usability of the existing system. We ran the study with five male subjects. One subject was a manager, one had expertise in organizational data, one was an expert in information visualization, and the final two subjects were computer science researchers. None of the subjects had previously used the system, although four had seen a demonstration of the system over a year before the study was performed. Each subject spent one and a half to two hours participating in the study, including 15 minutes to train IBM *ViaVoice*.

In the pilot study, we first explained to the subjects that the source of data to be analyzed was historical data taken from the POST database. This database is well-known to, and frequently used by, all participants of the study. Subjects were asked to articulate questions of interest with respect to analyzing such data for trends, and to articulate commands that they might want to use for interacting with infoVis views.

In the next part of the study, subjects spent some time using the system. They were given some instructions about the system, but were not given any formal training on using the system. At any point, the users could request help from the system by speaking a phrase such as: “help,” “please help,” or “what can I say?” At the end of the session, subjects reviewed their original list of queries and commands and articulated any modifications, additions, or comments.

5.2 Types of Questions Asked

Our current system supports the following types of questions about organizational data:

- How many people are in <ORG>?
- Where do people in <ORG> live?
- What do people in <ORG> do?
- How has <ORG> changed over time?
- Show all the new-hires for <ORG>.
- What locations did people come from?
- What organizations did people come from?
- List fields. [Returns the database schema.]

Table 1 summarizes the number of questions each subject asked and divides them into the following four categories:

- *currently supported:* questions that can be posed to and answered by the current system.
- *easily added:* questions that can be added to and supported by the system, given the existing data tables and views.

- *can be added with some work:* questions that require new data tables or alternative data views before they can be added to the system.
- *not supported by available data:* questions that cannot be added to the system, due to lack of information within the available data.

Table 1. Summary of Subjects' Questions About Organizational Data

	# of Organizational Questions				
Subject	**Currently Supported**	**Easily Added**	**Can Be Added with Some Work**	**Not Supported by the Available Data**	**TOTAL**
Manager	1	5	6	2	**14**
Org Expert	0	12	1	0	**13**
Vis Expert	2	6	2	0	**10**
CS Researcher 1	1	3	2	2	**8**
CS Researcher 2	2	3	2	1	**8**
TOTALS:	**6**	**29**	**13**	**5**	**53**

Each subject listed 8 to 14 questions of interest for a total of 53 questions. Forty-four of the fifty-three questions were unique (83%), with only five questions being articulated by more than one person. These five questions include the following:

- How has an organization changed over time?
- What are the variables that are recorded? [Equivalent to "List Fields."]
- How do organizations change parent organizations?
- What's been the turnover in an organization?
- Follow a particular individual over time.

The first two questions are currently covered by the system, the next two can be easily added, and the final question can be added with some work.

Our original collection of questions was produced by a single domain expert. The pilot study revealed that different users are interested in a variety of questions about organizational data, and that for complex data sets, there may even be little overlap in the questions that users do ask. Although our current system only supports a few of the subjects' articulated questions, our architecture is flexible enough to support all of the users' questions that can be answered by the available data. We plan to add such support in the future.

5.3 Usability of the Existing System

All subjects in our pilot study were able to use the existing system to find some data trends that they did not know about before interacting with the system. Some users also recognized tradeoffs between the speech and GUI interfaces with the system. Example cases in which they recognized speech was or could be more efficient or accessible than using the GUI include the following:

- commands for the data views—e.g., commands such as “select all” or “hide unselected,”
- selection based on data that may not currently be visible (e.g., “select new hires”), and
- selection that may be difficult or cumbersome to do with the mouse (e.g., “show me differences between bar A and B”).

Situations where they found the mouse to be more convenient include:

- clicking directly on a bar to select it, rather than saying something like “select bar 2,”
- moving windows (i.e., data views) around on the screen.

Although the subjects as a group recognized the utility of both interfaces, we did have subjects on each of the extremes of our multi-modal system—i.e., one who would prefer a mostly speech-only system and another who would rather have a GUI-only system. The manager thought that the system was “cool” and wanted to be able to interact with the system using only speech. He thought that it was relaxing to be able to sit back and speak commands to the system. On the other hand, one of the computer science researchers was frustrated by the speech interface. This subject was getting over a cold and was the only subject who could have used more than 15 minutes of training with *ViaVoice* to improve speech recognition. He also felt that the limited number of questions currently supported by the system could be provided directly with a graphical user interface, which he would have preferred.

Our mixed results in user preferences for interface modes indicate the need to broaden the access of user interactions via both interaction modes (natural language and GUI), and then to conduct new usability studies to further investigate users’ preferences within our multi-modal system.

6. DISCUSSION

The combination of the domain-independent and domain-specific aspects of our natural language interface provides the following advantages:

- *For large data sets, it increases the precision in data selection.* For example, it supports commands such as “select the top ten bars.” This is easy to say, but may be more cumbersome to specify with the mouse, especially for a large number of similarly sized bars.

- *A single voice command can encapsulate a sequence of InfoStill commands.* This is illustrated by the “sort by first and third columns”, which encapsulates two distinct InfoStill commands into one English sentence.
- *Users can provide partial information.* This can be seen with the “sort” command. Sisl recognizes this as a partial command, and conducts a dialogue with the user for the desired sorting criteria, such as by order or size.
- *It uses domain knowledge and an interactive design to guide users in exploring data.* The system conducts a dialogue with the user about his/her intended inquiries, refines these inquiries into complete and unambiguous database queries, and generates appropriate commands to send to InfoStill to create a set of relevant views. Users can thus explore the data without having to be familiar with the data set or with infoVis environments.
- *It allows users to choose the more natural mode of interaction for exploring data within an infoVis framework.* Users can explore the data and interact with views using natural language and standard direct manipulation. For example, they may use the natural language interface to have the system automatically create a set of relevant views, and then may use both the natural language interface and the traditional mouse and keyboard to manipulate these views.
- *The combination of InfoStill and Sisl provides a modular and extensible design.* In particular, it is straightforward to extend the Sisl grammar and logic to directly support other InfoStill commands, and to enrich the supported threads of inquiry about organizational data. These modifications can be done without requiring structural changes to the existing service logic and grammars, or to InfoStill. Conversely, InfoStill can be modified without requiring changes to the Sisl infrastructure. It is thus possible to easily extend the capabilities of the natural language interface based on feedback from users.

The main disadvantages of natural language interfaces stem from limitations of speech recognition technology:

- Automatic speech recognition technology (even state-of-the-art) is known to suffer from some inaccuracies. These inaccuracies increase with the size of the grammars and vocabulary that is recognized. Hence, it is infeasible to provide voice counterparts to all graphical user interface functionality. In fact, as observed in [11], such basic natural language interfaces often hamper users rather than aiding them.
- Some user activities are best done through a graphical user interface. For example, windowing commands are best done with a mouse; speaking commands such as “bring view 3 into focus” would be (at best) extremely tedious for users.

Our overall aim has been to develop natural language interfaces that can be used in conjunction with graphical user interfaces rather than to replace them. This keeps the size of the grammars tractable, while utilizing the benefits of natural language in order to provide domain specific guidance not available in traditional interfaces.

7. RELATED WORK

Previous work in adding an audio interface to visualization systems focuses on the use of audio as output or feedback to the user. That is, this previous work focuses on the use of sound to convey information. This mapping of numerical data to sound for the purpose of conveying information has been referred to as "sonification" [18]. Example applications of sonification to visualization include the use of sonification for scientific visualization [14] and for computational fluid dynamics [13]. In contrast to sonification, or the use of audio as *output* of a visualization system, we are focusing our efforts on the use of audio in the form of natural language speech *input* to an infoVis system. To our knowledge, this is a new input mode of interaction to an infoVis system that has not previously been explored.

Various interfaces have been developed to automatically generate visual reports of data, even including the generation of alert indicators with a set of reports, if appropriate (e.g., [12]). In these systems, expert data analysts script or code the report generation so that end-users can focus on reviewing key findings rather than on how to analyze the data. More specifically, end-users are relieved from needing to know details such as the format of raw data files or deciding which chart on what data is most useful for answering a particular inquiry of interest. The problem with this approach, however, is that end-users are limited to the set of inquiries provided, along with their corresponding hardwired solutions. In our approach, we address this issue through the use of the Sisl logic. That is, as we discussed in Section 3, the Sisl logic provides the flexibility to send commands that answer a genre of questions rather than a fixed list of questions. At the same time, like these other systems, the Sisl logic for generating InfoStill commands automatically maps end-user inquiries to relevant data subsets and views, thereby making data analysis within the InfoStill framework more accessible to novice or naïve users.

Perhaps the research most closely related to ours is the work on NL interfaces to databases (e.g., [9]). Beyond the early work in this area conducted in the 70's, more recent work has been conducted to examine the integration of visual, direct manipulation query languages to NL query interfaces (e.g., [2]) as well as the notion of multi-modal interfaces (e.g., [5]). Most of this work, however, has focused on accessing data more than analyzing data. Although Cohen's work [5] does examine the use of a multi-modal interface for some data analysis, his efforts focus on multi-modal query *input* whereas our work also supports the use of multi-modal manipulation of the *output* for exploring the data in search of trends. Overall, our work goes beyond database access to a more integrated approach of accessing the appropriate data and creating relevant views so that users can interactively analyze and explore their data through information visualization.

8. CONCLUSION AND FUTURE WORK

In this paper, we presented our approach to adding a novel speech and natural language interface to an information visualization framework for the purpose of aiding users in conducting data analysis. This interface uses domain knowledge and an interactive design to guide users in exploring data. Users can pose natural language inquiries about a particular data domain to the system and the system responds by displaying appropriate visualization views. If users give incomplete information, the system prompts and guides them in constructing an appropriate query. Once visualizations are displayed, users can interact with these views via natural language commands and standard direct manipulation, according to their preference.

Our system design uniquely combines three existing, yet powerful components through the use of simple API's. These components include: the IBM ViaVoice speech recognition system, the Sisl (several interfaces, single logic) architecture for structuring services with multiple user interfaces, and the InfoStill infoVis framework for aiding users in distilling information from raw data. We applied our approach to the analysis of some real-life organizational data, and we use examples from this first application to illustrate the utility of our design. The contributions of this work include the design of a novel, multi-modal interface to information visualization that empowers users to explore data in a way that has not previously been possible, guides them in their explorations when they need help, and allows them to choose the more natural mode of interaction for exploring data within an infoVis framework.

Our pilot study identified additional types of natural language questions to incorporate into our system. The study also showed that users with no training in our system could use it to find data trends, but that users had varied preferences of the interaction modes (multi-modal versus speech versus GUI). In the future, we plan to update our system with additional questions, broaden the capabilities of each interaction mode, and conduct additional studies on how users interact with the system to perform various data analysis and exploration tasks (e.g., to better understand when users select which mode of interaction and for what purpose). We also plan to apply our approach to analyze data from other domains, domains such as resource and network monitoring, and stock market trading.

REFERENCES

[1] Ahlberg, C. and B. Shneiderman. (1994). "Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays." *CHI'94 Conference Proceedings*. New York, NY: ACM Press, 313-317.

[2] Anick, P.G., Brennan, J.D., Flynn, R.A., and D.R. Hanssen. (1999). "A Direct Manipulation Interface for Boolean Information Retrieval via Natural Language Query." *Proceedings of the Thirteenth International Conference on*

Research and Development in Information Retrieval. NY:ACM Press, 135-150.

[3] Ball, T., Colby, C., Danielsen, P., Jagadeesan, L., Jagadeesan, R., Laufer, K., Mataga, P., Rehor, K. (2000). "Sisl: Several Interfaces, Single Logic." *International Journal of Speech Technology, 3*, 93-108.

[4] Carlis, J.V. and Konstan, J.A. (1998). "Interactive Visualization of Serial Periodic Data." *UIST'98 Conference Proceedings*. NY:ACM Press, 29-38.

[5] Cohen, P.R. (1992). "The Role of Natural Language in a Multimodal Interface." *UIST'92 Conference Proceedings*. NY:ACM Press, 143-149.

[6] Cox, K., Hibino, S., Hong, L., Mockus, A. and G. Wills. (1999). "InfoStill: A Task-Oriented Framework for Analyzing Data Through Information Visualization." *IEEE Symposium on Information Visualization Late-Breaking Results*.

[7] Conway, M. E. "How Do Committees Invent?" *Datamation* 14, 4, 1968, 28-31.

[8] Damper, R. and S. Wood. (1995). "Speech versus keying in command and control applications," *International Journal of Human-Computer Studies, 42*, 289-305.

[9] Hendrix, G.G., Sacerdoti, E.D., Sagalowicz, D., and J. Slocum. (1978). Developing a Natural Language Interface to Complex Data. *ACM Transactions on Database Systems. 3(2)*, 105-147.

[10] Hibino, S. (1999). "Task Analysis for Information Visualization." *Third International Conference on Visual Information Systems (VISual'99).* (Huijsmans, D.P. and A.W.M. Smeulders, Eds.). Berlin: Springer-Verlag, 139-146.

[11] Hugunin, J. and V. Zue. (1997). "On the Design of Effective Speech-Based Interfaces for Desktop Applications." *EuroSpeech'97 Conference Proceedings*.

[12] Knutson, J.F., Anand, T. and R.L. Henneman (1997). "Evolution of a User Interface Design: NCR's Management Discovery Tool (MDT)." *CHI'97 Conference Proceedings*. NY: ACM Press, 526-533.

[13] McCabe, K. and A. Rangwalla. (1994). "Auditory Display of Computational Fluid Dynamics Data." In (Kramer, G., Ed.) *Auditory Display: Sonification, Audification, and Auditory Interfaces (Proceedings ICAD'92, The First International Conference on Auditory Display.* Addison-Wesley, 327-340.

[14] Minghim, R. and A.R. Forrest. (1995). "An Illustrated Analysis of Sonification for Scientific Visualisation." *Proceedings Visualization'95 Conference*. Los Alamitos, CA: IEEE Computer Society Press, 110-117.

[15] Mockus, A., Hibino, S. and T. Graves. (2000). "A Web-Based Approach to Interactive Visualization in Context." *Advanced Visual Interfaces 2000 (AVI'2000) Conference Proceedings*. NY:ACM Press, 181-188.

[16] Molnar, K. and M. Kletke. (1996). "The impacts on user performance and satisfaction of a voice-based front-end

interface for a standard software tool," *International Journal of Human-Computer Studies, 45*, 287-303.

[17] Oviatt, S. and P. Cohen. (2000). "Multimodal Interfaces that Process What Comes Naturally," *Communications of the ACM*, *43(3)*, 45-53.

[18] Scaletti, C. (1994). "Sound Synthesis Algorithms for Auditory Data Representations." In (Kramer, G., Ed.) *Auditory Display: Sonification, Audification, and Auditory Interfaces (Proc. ICAD'92, the First Int'l Conf. on Auditory Display.* Addison-Wesley, 223-252.

[19] Seagate. Seagate Crystal Reports. 1999, http://www.crystalinc.com/crystalreports/

[20] The voice eXtensible markup language. VoiceXML Forum. http://www.voicexml.org

[21] Wills, G.J. (1995). "Visual Exploration of Large Structured Datasets." *New Techniques and Trends in Statistics*. IOS Press, 237-246.