Articulate: Creating Meaningful Visualizations from Natural Language

BY

Yiwen Sun
B.E., Shandong University, China, 2003

THESIS

Submitted as partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Computer Science
in the Graduate College of the
University of Illinois at Chicago, 2012

Chicago, Illinois

Defense Committee:

Jason Leigh, Chair and Advisor
Andrew Johnson
Barbara Di Eugenio
Luc Renambot
Tom Peterka, Argonne National Laboratory

ACKNOWLEDGEMENTS

First and foremost I would like to express sincere gratitude to my advisor Prof. Jason Leigh, for his seasoned guidance and continued support in my PhD study. I am deeply indebted to him for giving me broad freedom with patience throughout my research. This dissertation would not have been possible without his enthusiasm and motivation. My sincere thanks also go to Prof. Andrew Johnson. He was never too busy to discuss ideas or offer feedback. He consistently helped me along the research and study.

Besides, I would like to thank the rest of my dissertation committee: Prof. Barbara Di Eugenio, Dr. Luc Renambot and Dr. Tom Peterka, for serving on my committee and providing insightful comments.

I wish to extend my warmest thanks to all the faculty, staff and students at the Electronic Visualization Laboratory for their support. They provided such a stimulating and fun environment for me to learn and grow.

Lastly, and most importantly, I would like to thank my family. Despite the geographical distance, my parents always made sure I felt their confidence and encouragement. Their advice was always timely and useful to me. And I am extremely thankful for the immense love and support from my husband and soul mate Karl, who always believed in me even when I didn't. To them I dedicate this dissertation.

YWS

TABLE OF CONTENTS

TABLE OF CONTENTS (CONTINUED)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES (CONTINUED)

FIGURE PAGE

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
ENDURANCE	Environmentally Non-Disturbing Under-ice Robotic ANtarctic Explorer
GATE	General Architecture for Text Engineering
GUI	Graphical User Interface
NSF	National Science Foundation
NLP	Natural Language Processing
PCFG	Probabilistic Context-Free Grammar
SimVL	Simplified Visualization Language
SQL	Structured Query Language

SUMMARY

While many visualization tools exist that offer sophisticated functions for charting complex data, synthesizing various information and deriving insight from abstract data, they still expect users to possess a high degree of expertise in wielding the tools to create an effective visualization. Unfortunately the users of such tools are usually lay people or domain experts with marginal knowledge of visualization techniques. These users typically can't make use of the latest advances in visualization tools because of the high learning curve, instead they want to focus on the data, and often end up resorting to traditional bar chart or spreadsheet. To facilitate the use of the advanced visualization tools, I propose an automated visualization framework: *Articulate*. The goal is to provide a streamlined experience to non-expert users, allowing them to focus on using the visualizations effectively to generate new findings.

The 2008 National Science Foundation workshop report on "Enabling Science Discoveries through Visual Exploration" also noted "there is a strong desire for conversational interfaces that facilitate a more natural means of interacting with science." Even a decade ago this would have seemed far-fetched, but today there is renewed interest in the use of natural language as an interface to computing, with a variety of successful models that are able to understand the meaning of verbal descriptions in search engine, recommender system, educational technology and health applications. This inspired me to adopt a natural language interface in the automated visualization framework which would allow an end-user to pose verbal inquiries, and then let the system assume the burden of determining the appropriate graph, and presenting the results. It is hoped that such a capability can potentially reduce the learning curve necessary for effective use of visualization tools, and thereby expand the population of users who can successfully conduct visual analysis.

In this dissertation I present *Articulate*, an approach toward enabling non-visualization experts to leverage advanced visualization techniques through the use of natural language as the

primary interface for crafting visualizations. The main challenge in this research is in determining how to translate imprecise verbal queries into precise and meaningful visualizations. In brief, the process involves three main steps: first, extracting syntactic and semantic information from a verbal query; then applying a supervised learning algorithm to automatically translate the user's intention into explicit expressions; and finally, selecting a suitable visualization to depict the expression. While the initial prototype produces information visualizations, it will also be demonstrated how the approach can be applied to scientific visualization as well as the details of the user studies that validate the approach.

It is important to understand that the goal of this research is to explore how to automatically translate natural and potentially ill-defined language into meaningful visualizations of data in a generalizable way, in order to relieve the user of the burden of having to learn how to use a complex interface, so that they can focus on articulating better scientific questions in data exploration with the modern advances of visualization.

1. INTRODUCTION

1.1. Motivation

Nearly one third of the human brain is devoted to processing visual information. Vision is the dominant sense for the acquisition of information from our everyday world. It is therefore no surprise that visualization, even in its simplest forms, remains the most effective means for converting large volumes of raw data into insight. Over the past three decades, much has been investigated on the design of sophisticated visualization tools in a variety of disciplines. However, the effort end-users have to make to craft a meaningful visualization using these tools has been mostly overlooked. The users of such tools are usually lay people or domain experts with marginal knowledge of graphic design and visualization techniques. When exploring data, they typically know what questions they want to ask, but often do not know, or do not have the time to learn, how to express these questions in a form that is suitable for a given visualization tool, such as specifying a desired graph type for a given data set, or assigning proper data fields to certain visual parameters.

A 2008 National Science Foundation report "Enabling Science Discoveries through Visual Exploration" [Ebert08] noted that one of the main barriers hindering the adoption of advanced visualization tools is the steep learning curve associated with them. 2010 findings by Grammel [Grammel10] showed that novices to Information Visualization still tended to use traditional bar, line and pie charts over other chart types by more than 70% because of their familiarity with them. Modern visualization tools offer such an expansive array of capabilities that they can only be wielded by an expert trained in visualization. In some ways it's like expecting someone to know how to build a house by simply sending them to Home Depot. As producing visualizations remains a skilled and time-consuming task, researchers began to investigate computational approaches to simplify the crafting process. One option is to use a

declarative graphical language as the medium for visualization design. By allowing users to explicitly select the data fields, their relationship to each other and representative graphical markers, a precise graphical language specification for the visualization is formulated which directs the generation of graphics automatically [Mackinlay86, Roth94, Mackinlay07, Heer10]. Another approach is to streamline the creation of visualization through pipeline. By identifying relevant pipeline components and manipulating representing operators, a customized visualization application can be constructed automatically [Bavoil05, Santos09]. However, without knowledge of the underlying visualization components, it is difficult to understand which graphical markers to choose or what series of modules to be added to obtain an effective visualization. Consequently, the non-expert users still resort to the simplest tools, such as bar-charts and line graphs, even though they may lack the expressive power necessary to bring fundamental phenomena into focus.

Meanwhile, the 2008 NSF report noted “there is a strong desire for conversational interfaces that facilitate a more natural means of interacting with science.” In other words, scientists “simply” want to tell the computer what they want to see and have it just create it. They don’t want to have to become visualization experts. Zue [Zue00a] also pointed out that spoken language is attractive in interface design, because it is the most natural, efficient, flexible, and inexpensive means of communication. It is also a skill that humans have been using for several if not tens of millennia as compared to computers, which have only become widely available since the mid-80s. And while computing software and specific user-interfaces will become obsolete over time, the use of spoken dialog as the primary form of human communication is unlikely to become obsolete.

Even a decade ago this would have seemed far-fetched, but today I am seeing renewed interest in the use of natural language as an interface to computing. For example, survey results according to search engines like Ask.com show that a third of search queries are entered as natural language questions rather than keywords. Google's Voice Search allows a user to conduct web searches by directly speaking into their smartphones. Wolfram research in 2009 began

testing a service called Wolfram Alpha, which attempts to automatically visualize information based on natural-language fact-based queries. The field of natural language processing has made great strides in the last decades, with a variety of models that are able to understand the meaning of sentences in recommender systems, educational technology and health applications [Kaufmann07, Hallet08, Kersey09].

This inspired me to adopt a conversational interface for the automatic generation of visualization. A model like this would allow an end-user to pose natural language inquiries, and then let the system assume the burden of determining the appropriate graph, and presenting the results. It is hoped that such a capability can potentially reduce the learning curve necessary for effective use of visualization tools, and thereby expand the population of users who can successfully conduct visual analysis. However more importantly, by studying the problem of how to interface speech with visualization I am in fact attempting to understand the relationship between intent and visual grammar, and in so doing I may be able to begin to quantify the effectiveness of a visualization.

1.2. Contribution

In this research I am interested in exploring how to automatically translate natural, and potentially ill-defined, conversational language into meaningful visualizations of data in a generalizable way that enables users who are not visualization experts to make use of modern advances in visualization. Note however in this work it is not simply translating explicit visualization commands such as “make a scatter plot of X and Y”. I am attempting to produce visualizations from questions a scientist may ask about the data, such as “what is the correlation between temperature and depth with temperature below zero”. A key benefit of this approach is that it enables the end-users to focus on the scientific question they are trying to ask rather than the specific operations that must be manipulated to produce a representative visualization.

The main contribution of this dissertation is the proposed methodology for translating imprecise natural language queries into meaningful visualizations. To be specific, an imprecise query is represented by a feature vector, and classified as certain visualization task, then translated to precise commands and presented automatically as a graph. This methodology can be expanded to different domain science visualizations by adjusting certain steps in the framework. The novel design of this methodology is three-fold:

First, the incorporation of a speech interface and natural language parser enables the user to issue requests without conforming to a rigid grammatical structure. This aims to relieve the user of the burden of having to learn how to use a complex interface, so that they can focus on articulating effective scientific questions and generating new findings.

Second, the introduction of a meta-learning algorithm towards automatically interpreting a user's intent based on linguistic features. As Wilkinson [Wilkinson05] pointed out "much more difficult is to understand what users intend to do with their data when making a graphic. Instead of taking this risk, most charting packages channel user requests into a rigid array of chart types". To avoid the rigidity and accommodate different visualization purposes, a multi-dimensional feature space is defined to capture a variety of syntactic and semantic features in users' verbal description. Based on it, the verbal description can be represented by a feature vector, and classified as a specific visualization task, then translated to precise commands automatically. Here the classification algorithm is derived from the results of three base-learners to avoid the bias from a single learner.

Third, the capability of suggesting related queries and visualizations. Besides the primary recommended visualization, a list of "next-best" candidates are provided to the user, which take into account their previous preferences. Such context-aware suggestions can potentially help them consider alternative perspectives on the data that they may not have originally envisioned.

1.3. Document Structure

The remainder of the dissertation is organized as follows. I begin the discussion of related work with Chapter 2, including an overview of approaches for automatic generation of visualization, the declarative visual specification languages and various visual task taxonomies, as well as the evolution of the language interface. In Chapter 3, I conduct design space analysis at three different levels, which set the stage for the main focus of this work - the proposed framework of *Articulate.* This framework includes three essential components: Data Parser, Input Translator, and Visualization Executer. The methodologies underlying each component are explained as well. The implementation details and the intrinsic evaluation of *Articulate* are presented in Chapter 4. In Chapter 5, I provide the details of the user studies that validate the approach. While the initial prototype produces information visualizations, I will also demonstrate how the approach can be applied to scientific visualization. Finally, I conclude this work in Chapter 6 by summarizing the present research results and outlining directions for future work.

2. RELATED WORK

This chapter presents the prior and ongoing work pertaining to the design of automated visualization systems. It begins by providing a brief overview of the evolution of systems that support the automatic generation of visual representations. Following is the review of various declarative languages used in visualization specification. Then several taxonomies of visualization tasks are presented at different levels of abstraction to help characterize users' intention. Lastly, current advances in the speech and natural language interfaces are discussed which motivated me to consider the use of natural language as a primary interface for crafting visualizations.

2.1. Automatic Generation of Visual Representation

Two decades ago researchers began exploring approaches for the automatic generation of visual representation. One of the early piece of work was Mackinlay's APT system [Mackinlay86]. It introduced a composition algebra to describe various graphical encoding and developed expressiveness and effectiveness criteria to ensure the meaningful design. The SAGE system [Roth94] extended the concepts of APT, providing a richer set of data characterizations and generating a wider range of composite views by interaction. The previous work on automatic presentation focused primarily on single views of data; however, Show Me [Mackinlay07] provided support for small multiple views. It included an integrated set of user interface commands and defaults that automatically generate small multiple views based on VizQL – an algebraic specification language. Users place data fields into columns and rows in the interface panel to specify VizQL commands. In order to generate insightful visualizations, an understanding of the relationships between columns and rows is needed.

While above work focused on identifying and encoding data in discrete graphics, there is another trend in addressing the issues of communicating with users in visual discourse. Feiner's Apex system [Feiner85] set the foundational work in this area. It attempted to automatically create visual discourses - a series of animated visual illustrations for explaining complex information to users. And his work was extended in IMPROVISE [Zhou98], which used an AI planning-based approach to automatically design and create such discourses.

In recent years, related approaches have targeted specific application domains, rather than proposed a more general methodology. Kerpedjiev and Roth introduced AutoBrief [Kerpedjiev01], a system that automatically summarizes transportation schedules as text and graphs. In this system, they proposed a set of rules to map communicative goals to low-level operational tasks, such as lookup or compare. However, the system only focused on generating explanations of problems and relations existing in the data but not in response to user's ad-hoc requests. Gilson et al. [Gilson08] proposed an approach for automatic generation of visualizations via ontology mapping and applied the approach to web data for music. The web data was first mapped to domain ontology, and then projected to visual representation ontology, which was finally depicted as a specific visualization using external visualization toolkits. The mapping between domain and visual representation ontologies was represented by semantic bridging ontologies, which were defined from expert knowledge. By comparison, my approach uses a more flexible meta-learning algorithm to automatically translate language into visualization intentions. Rezk-Salama et al. [Rezk06] demonstrated a semantic model for automatic transfer function editing in volume rendering applications. However the design of the semantic model is such that the presence of computer scientists and domain experts are needed when selecting meaningful semantic parameters and mapping parameters to low-level primitives. Another interesting approach is VisMashup [Santos09], which simplified the creation of customized visualization applications with a pipeline. Once the pipeline is constructed, the application can be generated automatically. While this infrastructure enables an application designer to assemble

custom applications quickly, the designer still needs some visualization background to build up the pipeline from components or templates. Although *Articulate* shares some of these same goals, it goes a step further by allowing the users to verbally articulate what they want to see with minimal apriori knowledge of how to use the user interface.

Recent work in the information visualization community has applied various design principles to the automatic generation of visualizations though none had used natural language nor approached scientific visualizations. *Articulate* attempted to combine these advanced techniques together in exploring how to automatically translate natural, and potentially ill-defined, conversational language into meaningful visualizations of data in a generalizable way that enables users who are not visualization experts to make use of modern advances in visualization.

2.2. Declarative Language

Declarative Language has become of particular interest recently, due to its simplicity and generality in programming paradigm, which describes what computation should be performed and not how to compute it. Thus, it is easy to understand and has been successfully applied to a wide array of different real-world situations. For instance, SQL (Structured Query Language) allows the user to describe desired data in query statements, leaving the database management system responsible for planning, optimizing, and performing the physical operations necessary to produce that queried result. And markup languages such as HTML and XML enable novice programmers to develop web pages and exchange data over the Internet. Researchers in the visualization community have also begun to note the benefits of declarative languages. Wilkinson‘s Grammar of Graphics [Wilkinson05] abstracted the process of putting data onto statistical graphics; ggplot2 [Wickham09] implemented the Grammar of Graphics in the R language. Stolte et al. [Stolte02] extended Wilkinson’s idea into VizQL, a specification language that was devised to describe the structure of a visual presentation as well as the data queries that

populate the view, which has been incorporated into a commercial visual analysis system called Tableau. Inspired by the powerful declarative languages like HTML/CSS, Bostock and Heer proposed Protovis [Bostock09, Heer10], a declarative language embedded within a host language like JavaScript, for constructing interactive visualizations accessible to the web. While all these approaches concentrate on 2D information visualizations, the only grammar I have been able to find for scientific visualization has been Duke's [Duke09] work in the creation of domain specific languages embedded in Haskell – a functional programming language for producing scientific visualizations. Duke's work shows how declarative languages enable rapid exploration of advanced visualization techniques; however, it lacks the comprehensiveness of the Grammar of Graphics.

2.3. Visualization Task Taxonomy

Recent findings by Grammel [Grammel10] showed that during the visualization construction process, there are three central activities: data attribute selection, visual template selection and visual mapping specification. The major barriers faced by information visualization novices were translating questions into data attributes and designing visual mapping that support answering these questions. Part of the solution they derived is to support automatic translation and mapping. The challenge of the automatic translation is to capture user's logical rationale behind each activity. However, most existing visualization systems are event-based systems that are designed to recognize and process low-level user interaction events like mouse clicks and drags, but can rarely understand and capture the semantics of user's activities. To capture the high-level semantic nature of a user's intention, I first examined the current research in the area of visualization task taxonomy.

A number of taxonomies have been developed that characterize visualization tasks. Shneiderman [Shneiderman96] sort out numerous tools and identify a principle of task taxonomy in a broad scope, which includes seven basic tasks that visualization systems need to support: *overview, zoom, filter, details-on-demand, relate, history, and extract*. This task taxonomy is like a streamlined process following the visual-information-seeking mantra "Overview first, zoom and filter, then details on demand". Wehrend [Wehrend90] proposed a classification of visualization based on a matrix formed by two crossed classifications: object and operation. The objects are categorized based on the nature of things in the target domain, such as scalar, direction, and position. And the operations attempt to distinguish users' different viewing goals, including *identify, locate, distinguish, categorize, cluster, distribution, rank, compare, associate, correlate*, etc. A problem with such classification is that it is grounded by common object classes, and based on the object classes to find the appropriate visualization operation, which could be an issue for complex objects like multi-fields or structure. Also their wide variety of operations almost exhausting of all the possible operations may not be applicable to automated design. Similar to Wehrend's taxonomy, Amar et al. [Amar05] have come up with ten basic task types to describe various users' needs in information analysis: *retrieve value, filter, compute derived value, find extremum, sort, determine range, characterize distribution, find anomalies, cluster and correlate*. The limitation of this taxonomy is that it mainly focused on the analytic activities. The above taxonomies characterize high-level human cognitive tasks or visual perceptual behaviors. In contrast, Gotz [Gotz08] distinguished users' visual analytic activity at multiple levels of granularity: task, sub-task, action, and event. They uniquely divided the action tier into three categories: *exploration actions, insight actions, and meta actions*, which can be used to both assist in inferring higher-level subtasks, and as a guide for defining new action types. Inspired by these taxonomies, I derived a more abstract categorization of visualization tasks, specifically tailored to characterize users' behavior in the natural language guided environment. It provides a

common vocabulary for expressing user's semantic intention and facilitates the automated translation of it. The details of this taxonomy will be discussed in Chapter 3.

2.4. Speech and Language Interface

A decade ago the possibility of widespread use of speech interaction seemed far-fetched, on both cognitive and technical reasons. Shneiderman [Shneiderman00] argued that speech input has limited value in human-computer interaction except in niche applications - such as for the disabled or answering service systems. The key criticism cited was that problem-solving and recall competed with speech articulation and interpretation in their use of working memory. However, Dennett [Dennett92] argues that problem solving is in fact enhanced when more areas of the brain are engaged such as when you are speaking and hearing your own words (i.e. thinking a problem out loud). Hanks [Hanks10] argued that "Humans are social animals, and language is the instrument of their sociability, as well as the vehicle of their thought processes." Language interfaces in a variety of settings, from educational technology to health care, have been shown to improve the user's experience and performance [Grasso98, Hallet08, Kersey09, Schulman09].

Technically, the considerable renewed interest in the use of speech and natural language as an interface to computing is due in large part to significant computing power and new powerful statistical models that are brought to improve speech recognition and natural language interpretation. For example, Ask.com reported that one third of web queries entered by users are now in the form of natural language rather than as keywords, and Google's director of research Peter Norvig, believes that being able to converse with computers is "the future" [SRN09]. Siri - the intelligent personal assistant on iPhone 4S allows users to send messages, schedule meetings, place phone calls, conduct web searches by directly speaking into their smartphones. NLP, the processing of language beyond the recognition of words in speech, also made great strides in the

last decade, with a variety of models that are used to understand the meaning of sentences, as shown by successes such as that of IBM Watson which defeated the two best human champions in Jeopardy! [Watson11], and Wolfram Alpha [Wolfram] - a knowledge engine developed by Wolfram Research that is capable of responding to natural language based questions with computed answers and relevant visualizations instead of a list of web pages as a traditional search engine provides. Additionally, a number of speech-based computational models have been developed that help users to access information using a conversational paradigm. JUPITER [Zue00b] for example allows users to obtain worldwide weather forecast information over the phone using spoken dialogue. It is a mixed initiative system [Zue00a] that requires zero user training, and accepts a large range of user inputs. This approach has been extended to similar domains where the vocabulary is sufficiently limited to support practical conversational paradigm, such as travel planning [Seneff00], health information access [Sherwani07]. But few of the systems I surveyed targeted the problem of visualization.

Cox et al.'s work [Cox01] was the first to integrate a natural language interface into an existing information visualization system. Cox's work takes the natural language question, determines the appropriate database query, and presents the results to the user. Results from a pilot user study indicated a considerable need for natural language interface to aid users in data analysis. However, the number of questions supported by the system was small and mostly grammar dependent. More importantly, in the result generation phase, they did not take the advantage of varied plot types based on the characteristics of different analytic tasks, but instead only utilize tables and bar charts.

These works inspired me to adopt a natural language interface for the automatic generation of visualizations. This design addresses a growing trend – the need toward a more natural means for interacting with the exponentially available data.

3. DESIGN AND METHODOLOGY

This chapter describes in detail the methodology of using natural language as a primary interface for crafting visualizations. By studying the problem of how to interface speech with visualization I am attempting to understand the relationship between *intent* and visual grammar, and explore how to automatically translate natural, and potentially ill-defined, conversational language into effective visualizations of data in a generalizable way.

The first part of this chapter discusses the design principles, from high level interaction modality, to intermediate level intention recognition, and then to low level visual display implementation. The design principles set the stage for the main focus of this dissertation - the proposed framework of *Articulate.* This framework includes three essential components: Data Parser, Input Translator, and Visualization Executer. The approaches underlying each component are explained as well. In brief, the approaches involve: first, extracting syntactic and semantic information from a verbal query; then applying a supervised learning algorithm to automatically translate the user's intention into explicit commands; and finally, determining an appropriate type of visualization based on the translated commands, the meta-data information and the impact of various graphical types.

3.1. Design Space Analysis

There has been a huge interest in designing visualization systems that can generate meaningful visual representations automatically. Meanwhile, there is an increasing desire for adopting natural language interface for its speed and ease of use. In this work, I seek to combine these two trends together, to investigate the question of how to design an intelligent visualization system that understands user's natural language questions and provides meaningful results promptly.

To conduct such a design, I examined various examples in the related fields (as discussed in Chapter 2), and identified important design features at three different levels: interaction level, intention level, and visual display level.

3.1.1. Interaction Level

The first and highest level is the user's interaction modality, which refers to the way a user communicates to the system. A much less examined modality for interacting is by speech. Speech interfaces are most effective when recognition accuracy is high, when they do not require the user to memorize a vocabulary or grammar, and when it takes less time to utter the verbal command than to perform the action through direct manipulation. For a speech or language interface, the input can be divided into three categories: predefined commands, natural language questions and human-like conversational dialogues. For instance, "make a scatter plot of X and Y" is an example of an explicit visualization command with two parameters. As shown in the example, command language has simpler syntax and is easier to parse. But it requires additional effort for the user to learn and translate from his or her mental model to the defined language model. Furthermore, the balance between simplicity and expressiveness is an issue the designer of the commands has to consider. To reduce the learning curve for novice users, adopting natural language as the input is a choice. It makes it easier to express their intent and allows more freedom for the user. In this dissertation, I am interested in the questions a scientist may ask of the data, such as "*what is the correlation between temperature and depth with temperature below zero*". By applying natural language processing techniques rather than keyword match to such kind of input, the system could capture important linguistic features of the input, get a better understanding of the intention of the question, and potentially provide a more relevant answer to the user. However, the flexibility of natural language brings in another challenge -- the ambiguity

of the input. This issue can be avoided to some extent with the help of domain-specific knowledge. For example, if it is known that certain meanings of a word will not occur in a specific domain or context, it can be eliminated from the dictionary of entity keywords. A more complex input style is the human-like conversational dialogue, which involves many referring expressions and moves in context. Here is an excerpt of the conversation between a visualization expert (VE) and two domain scientists (SCi) about a system that visualizes time series observations of zebras:

SC1: What are we seeing here?

SC2: How do I turn it just a little bit towards me?

VE: Use the right mouse button and then do this.

This excerpt includes many referring expressions, such as *it, here, this*. A dialog-language interface needs to resolve references to know what commands to execute. Understanding such complicated references requires a dialog model with advanced linguistic and domain knowledge analysis. In this dissertation I will focus on simpler style of natural language questions, since developing a system that can fully engage in a conversation with its users is beyond the scope of my research.

3.1.2. Intention Level

The second level in the design space is the intention level. It conveys the user's main objective behind an utterance, whether it is a command, a request or a follow-up. The intention level serves as an intermediate between those high-level input and low-level visualization. Here are a couple of examples of users' utterances and their underlying objectives:

"*Show me the relationship between pH and depth*" (a request to produce visualizations)

"*Add country to y axis*" (a command to modify the existing visualization)

"*Show the conductivity value when depth equals 7.5*" (a request for information about specific data values)

"*What are the other attributes around depth of 7.5*" (a follow-up from the previous request)

"*Can this dot represent an individual*" (a request for clarification on what the visualization represents)

The purpose of identifying the user's intention is to distinguish problems for which the user's goal in viewing the representations differs. In Abela's chart chooser [Abela81], he classified the function of a chart presentation into four categories: comparison, relationship, composition and distribution. But this approach only covers the situation of an initial generation of a chart. As reviewed in the related work, Shneiderman identified a principle of task taxonomy, which includes seven basic tasks that general visualization systems need to support: *overview, zoom, filter, details-on-demand, relate, history, and extract*. Although it is straightforward to find language equivalents for all these tasks, it is noticed some of them are simple to achieve via GUI but difficult to articulate aloud. For example, a continuous zooming-in operation on a 2D plot is easy to complete using mouse drags, but cumbersome with language. On the other hand, there are tasks that could be handled easily with a natural language interface, but difficult to specify with GUI. For instance, a query with various constraints, like "*what are the American cars get over 20 mpg*" is simple to say, but can be clumsy to perform with the mouse. In addition, there are cases when the context is easily maintained within the discourse, like the example shown in Figure 12. Guided by the above observations and Shneiderman's task taxonomy as well as Abela's chart chooser, I specifically tailor the taxonomy of user's intentions for visualization tasks into four groups to characterize users' behavior in natural language environment (as illustrated in Figure 1):

Sketch tasks are performed when users want to get a brief general presentation, usually with an intention of gaining an overview of the data. It is like the established shot in filmmaking, which introduces the context and shows the relationship between important subjects. Most tasks

fall into this category, for instance comparison task is primarily intended to compare values over time or categories by means of a time-series plot, bar chart or area chart. Relationship task focuses on discovering the correlation between two or more attributes. A scatter plot, bubble plot, radar plot or parallel coordinated plot is often used for this task. For data specified in XML-based GraphML format or TreeML format, such as people who know each other in a social network, or web pages that are connected to each other, the relationship graph can reveal the structure or interconnection among all the data entities in a tree or mesh diagram.

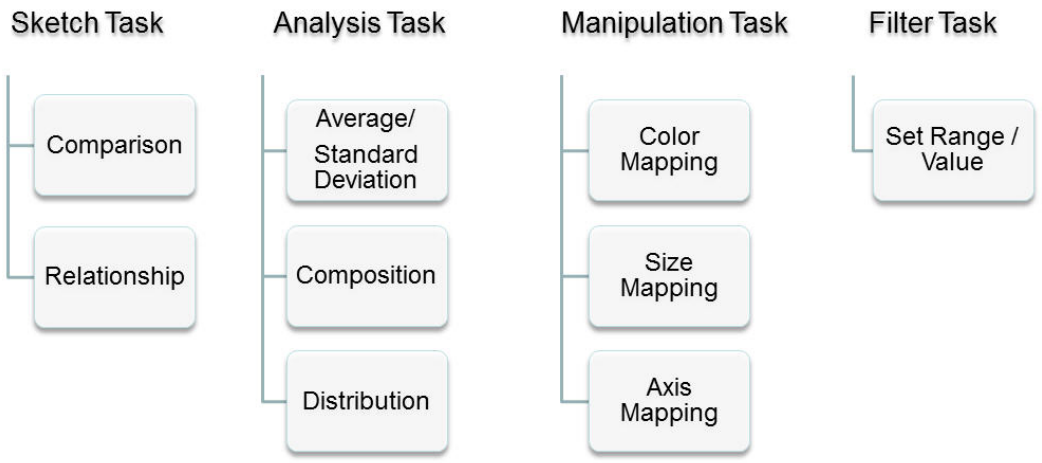

Figure 1. The classification of user's intentions in visualization tasks.

Analysis tasks consist of tasks that involve the use of statistical methods to summarize the features or patterns of a sample data by describing what was observed in the sample numerically. Examples of the numerical descriptors include: *mean or standard deviation*, a quantity calculated from a set of continuous data; *composition or percentage*, a quantity intended to reveal the structure or combination of the categorical data, for which often used visualization

methods are tree-map, pie chart; *distribution*, a quantity often intended to illustrate relative magnitudes or frequencies.

Filter tasks allow users to specify an interesting set of data, such as a particular value or value range of an attribute, and filter out unintended items, like a selection statement in the database operation. Once the data has been trimmed, users can quickly focus on their interests. Users' intensive usage of filter tasks was discovered in the preliminary study [Sun10]. The study showed that queries like "*what are the other attributes around depth of 7.5*", "*what is the heaviest American car*" are often encountered as follow-ups from sketch tasks.

Manipulation tasks describe ways to alter an existing visualization, such as remapping colors to data attributes, or switching axes. It is also noticed there are certain kinds of manipulation tasks, like zooming in or out, which are simple by direct manipulation with a mouse, but difficult to articulate aloud. Hence, I intend to avoid wasting effort on trying to provide natural language equivalents for these tasks.

3.1.3. Visual Display Level

The third level in the design space is the visual display level. It translates user's intentions into precise visual representations. As the famous saying goes, "a picture is worth a thousand words". Visual representations can translate large amounts of abstract information into a visible form that make it easy for users to perceive salient aspects of the data intuitively. There is tons of visual representations available ranging from 2D points, lines to 3D volume. Each type has its own advantages and disadvantages for certain kinds of data. For example: bar charts are often used for comparing values by category, whereas pie charts are good at illustrating relative magnitudes of parts to a whole. Harris [Harris00] provided a good overview of different kinds of chart. The big challenge is to choose the right one to resolve user's request. And the difficulty is that there is not yet an established theory on the effectiveness of different graph types or visual

representations. But many researchers have attempted to provide design principles. Cleveland and McGill's experiments and observations [Cleveland84] provided a guideline for the construction of statistical graphics. They first identified a set of elementary visual primitives (position along a common scale, position along non-aligned scales, length, direction, angle, area, volume, curvature, shading and color saturation), and then ranked them based on the accuracy with which people can extract information. According to their findings, the order from most to least accurate is: 1) position along a common scale, 2) position along non-aligned scales, 3) length, direction, angle, 4) area, 5) volume, curvature, 6) shading and color saturation. For example, in the task of quantitative judgment, the visual perception of length is more accurate than area, which in turn is more accurate than volume. Tufte [Tufte83] also addressed several important principles in the visual display of quantitative information, such as: "above all else show the data"; "maximize the data-ink ratio"; "graphics should tend toward the horizontal, greater in length than height". Inspired by these visual display principles, a set of rules are derived for determining an appropriate visual representation in this framework.

First, the properties of the graph type must match the property of the data (such as the number of variables, data types, and whether the variables change over time). For instance, in the query "*how has the price of beef changed over years*", the user is interested in a time-series variable; thus a time-series plot, often a line chart, is preferred than other charts, in which an x dimension representing the regular ordering of the time scale makes this graph efficient to interpret.

Second, pre-processing the data when necessary to reduce cluttering and facilitate the perception of the graph. Take "*what is the distribution of CDOM*" for example; the user's intention is to see the distribution of the data attribute CDOM, expecting any pattern or anomalies in the spread of data. But the actual value of CDOM ranges from 2.3 to 45 with over 400 unique values. Counting each value as a unique interval in the histogram will lead to too many bars and flat out the graph. Hence, it is necessary to define a limited number of regular-sized bins, and

partition the raw data into the bins, to make the distribution pattern more salient. Details of the data processing or transformation methods will be discussed in Section 3.2.3.

Third, encoding more important information with more effective visual primitives. Here I am focusing on the visual primitives of position, length, area/volume and color. For instance, in the presence of multiple variables such as "*how does GDP change over years by country*", encoding "GDP, Year and Country" with "position, position and color" will be more suitable than "color, position, position", because GDP is the most interesting variable to the user. Another example is "*can you compare the price of beef with eggs*", where the user intend to compare two quantitative attributes. Putting them on the y-axis of a line chart, or using the length of bar as indicator will be easily perceived than showing it in a pie chart, because the perception of the position or the length on a common scale is more accurate than the area.

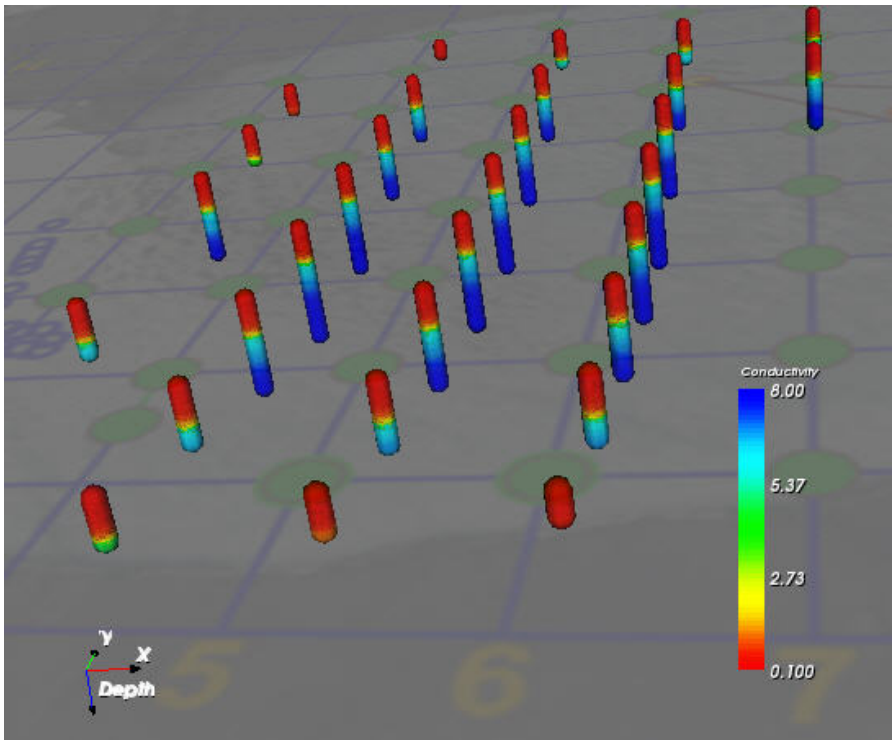

Figure 2. 3D visualization for the query "*show the relationship between conductivity and depth with regard to X and Y*".

Although these principles are mainly derived from the visual display of 2D chart, most of them are also applicable to 3D scientific visualization. For example, in the multivariate query "*show the relationship between conductivity and depth with regard to X and Y*". *X, Y* represent positions on the x and y dimension; *depth* is visualized as the third dimension naturally. Then another visual primitive has to be chosen to depict the variable of *conductivity*. In this case, color is a better choice, because it is more prominent than length or area/volume when the data points are very closely located in the three dimensional environment (as shown in Figure 2).

This design space analysis highlighted several principles in the design of the natural language guided visualization system. Through the analysis, I identified four distinct categories of user's intentions to help translate the high-level specification into low-level visual display implementation. And a set of rules was derived in the implementation of effective visual representations. These results guide the following framework design practice.

3.2. Framework Design

Based on the design space analysis discussed above, I developed a generalized framework – *Articulate*, which is able to understand a user's imprecise request, translate it to precise intention expression, and generate a purposeful visualization accordingly. Figure 3 outlines the major components of the framework. Throughout *Articulate*'s development, I performed a couple of pilot studies to gauge the approach's potential and discover which aspects were in need of improvement. The current design is the result of an iterative design process that has addressed many issues that were identified by preliminary users.

To illustrate how the model works, let us look at an example. Suppose a user loads in a dataset about hydrologic information, and makes a query: "*How was conductivity related with*

depth". The Data Parser will first read the original data file collecting information such as attribute names, data types, and matching the words *conductivity* and *depth* to existing attributes. Meanwhile, the Input Translator will interpret the sentence as a sketch task; more specifically a request for producing a relationship graph with two attributes assigned as the x and y axes. This interpretation is expressed as a series of SimVL commands and entered into the Visualization Executer, where information about the data from the data parser and the intermediate commands are funneled. Properties of various graph types are also funneled in via the graph specification. The Visualization Executer uses this information to determine that the most appropriate visualization will be a scatter plot.

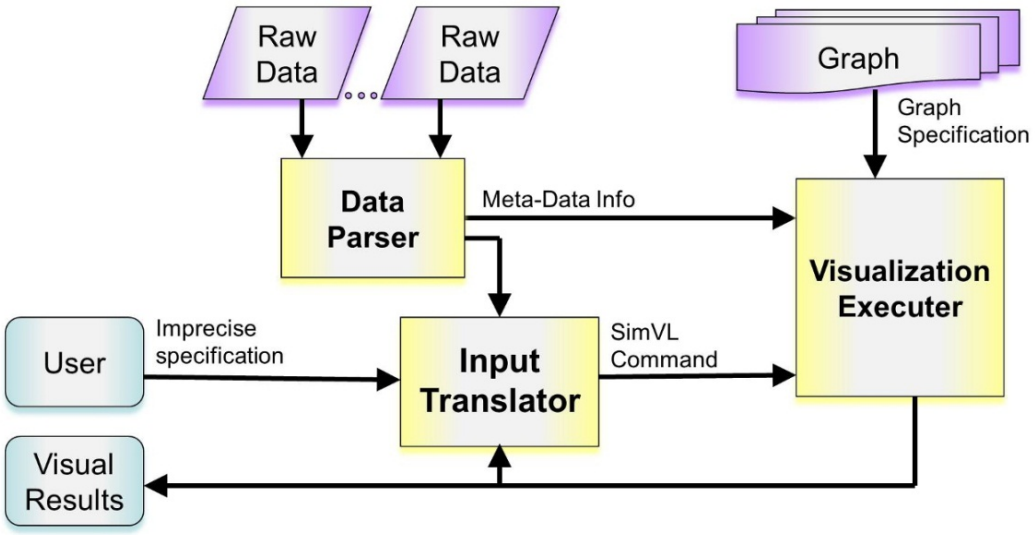

Figure 3. Overview of the *Articulate* System.

There are three essential parts of the framework: Data Parser, Input Translator, and Visualization Executer. In what follows, I present the methodologies underlying each component.

3.2.1. Data Parser

The Data Parser will first read the raw data collecting information such as attribute names, data types (numerical values, text or time) and data structures (tables or trees). Furthermore, meta-data properties are gathered such as data units and semantic relations, which provide a brief context for interpreting the data. This meta-information is then sent to the Visualization Executer to help determine the most appropriate graph to produce. This subsection will discuss how this meta-information is derived.

Ideally when articulating a query, the user will mention an attribute name as it appears in the data file. But in reality this is not always the case, as discovered in my earlier study (Section 5.1). For example, in a dataset regarding average food prices, data values may include chicken, beef and pork. However users posed queries such as: "*compare the price of meat*". Clearly, searching for the exact data value in the query will not lead to a desired result. To help recognize such terms, the following meta-data are utilized:

- Surface word: the original attribute name. Here, each attribute name is often regarded as a special property noun and usually appears as a column in the data file.
- Data type: number, text, date / time. This information is used in Visualization Executor to determine a specific visualization type. E.g., year might suggest a time-series plot as the appropriate visual representation.
- Data Units: units are useful in determining which attributes can share an axis in queries such as "*compare the price of apples with tomatoes*" versus "*compare cylinders with horsepower*". In the first example, apples and tomatoes have the same data unit (price), so putting them both on the y axis and using year as the x axis is

more desirable than plotting them on x, y axes respectively. In the second example however, the two attributes cylinder and horsepower have unmatched units, and therefore a user would expect them plotted on the x and y axes separately.

- Semantic relations: A semantic relation reflects the relation between concepts and meanings used in the language. Here I include four important kinds of semantic relations in the meta-data:
  - Synonym - a list of words with almost identical or similar meanings to the attribute name. For example, "United States" is a synonym of "America".
  - Hypernym - the generic term used to designate a whole class of the specific attribute. For example, "meat" is a hypernym for "beef".
  - Holonym - a term denoting the whole, as opposite to a term denoting a part of, or a member of the whole. For example "automobile" is a holonym of "wheel".
  - Pertainym - A relational adjective, which is usually defined as "of or pertaining to" a noun or other pertainym. For example, "Japanese" when used as an adjective pertains to "Japan".

To obtain the semantic relations, WordNet [Miller95] is used, which is a lexical database that groups English words into sets of cognitive synonyms (synsets) and expresses various semantic relations between these synsets. In this way, I can expand each attribute into an insightful meta-word and send that to the Visualization Executer to help determine the most appropriate graph to produce.

3.2.2. Input Translator

The Input Translator takes an imprecise natural language query spoken by the user and interprets its intention as a precise request for a specific visualization task. The result of the interpretation is expressed as commands that follow a formal grammar, which I call SimVL

(Simplified Visualization Language). As the core component in the system framework, the Input Translator performs a sequence of fundamental operations to smoothly connect the high-level input and low-level visual representation together. Figure 4 demonstrates the major steps in the workflow. The approach of each step is detailed in the following sub-sections. As illustrated in Figure 3, the Input Translator is also aware of previously created visualization and adjusts its decision making and alternative suggestion based on that.

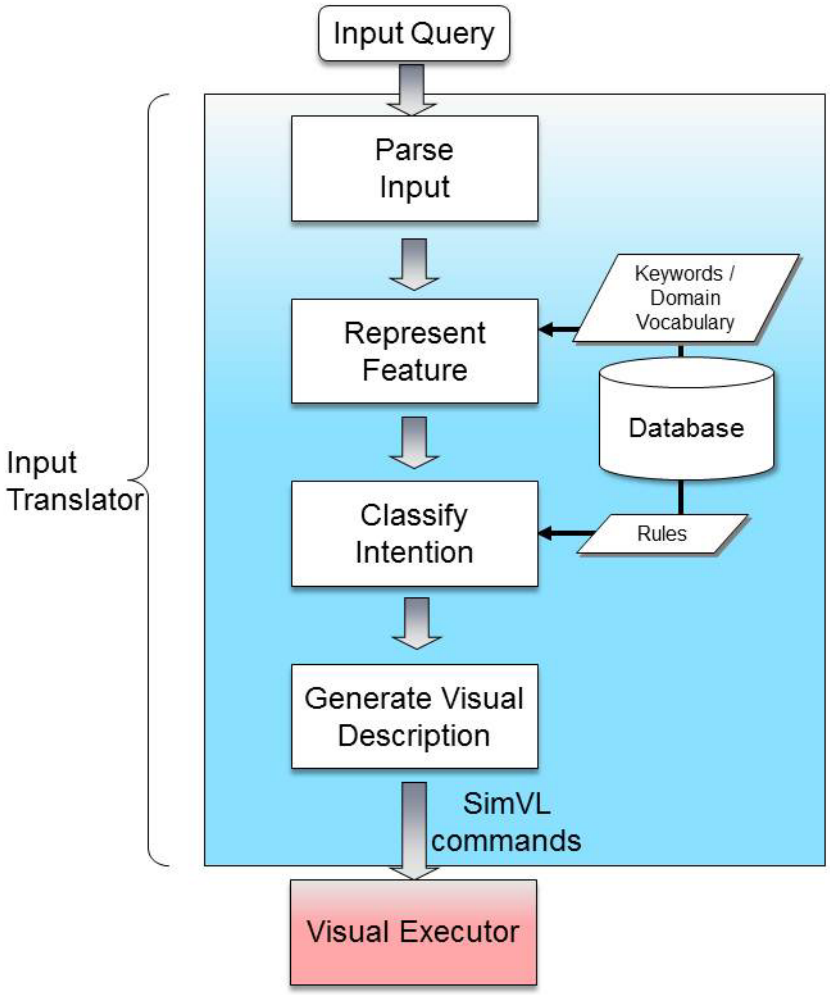

Figure 4. Basic workflow in the Input Translator.

The output of the Input Translator is supplied to the Visual Executor in the form of SimVL commands, which include two essential pieces of information translated from a user's query: data of interest and intended tasks. The process of recognizing the target data attributes from a user's specification is based on natural language parsing, which is explained in Section 3.2.2.1; while determining visualization intentions is made automatically by a meta-learning algorithm, which is discussed in Section 3.2.2.3.

3.2.2.1. Parsing the Input Stream

User's initial input to the system is a stream of text representing the natural language query. The query sentence is first parsed into a sequence of words tagged with part-of-speech labels using the Stanford Parser [Klein03]. These labels marks the lexical category for each word, such as noun, verb, adjective, adverb, preposition, and etc. based on both its definition, as well as its context. For example, the sentence "*How does conductivity relate to depth when depth equals 7.5*" will be tagged as:

How/WRB does/VBZ conductivity/NN relate/VB to/TO depth/NN when/WRB depth/NN equals/VBZ 7.5/CD

Using these tags, the different functions of each word can be distinguished. In addition, the stem of each word, i.e. the root of the word, is also extracted. The stemmed result for the previous example is shown below:

how do conductivity relate to depth when depth equal 7.5

Compared with the original sentence, the difference is all about the tense of verbs: "was" is stemmed as "be", "related" is stemmed as "relate". The stems avoid the morphological ambiguity. For example, relate, relating, related all have the same root, and should be recognized as the same keyword in the classification step.

The syntactic structure is also extracted from the query sentence, and expressed in a parse tree where leaf nodes store the words and internal nodes show the part-of-speech tags or phrasal labels (Figure 5). This tree provides the structure of the sentence, such as, which groups of words go together, for example "*depth equals 7.5*" forms a simple declarative clause as denoted by their parent node S. And additionally, typed dependencies [Marneffe06] are obtained to describe the grammatical relationships between pairs of words in triplets: name of the relation, governor and dependent. It clearly explains which word is the subject or object of a verb. The following are the dependencies for the previous query.

advmod(relate-4, How-1)
aux(relate-4, does-2)
nsubj(relate-4, conductivity-3)
root(ROOT-0, relate-4)
prep_to(relate-4, depth-6)
advmod(equals-9, when-7)
nsubj(equals-9, depth-8)
advcl(relate-4, equals-9)
dobj(equals-9, 7.5-10)

Using these structures, the functions of each word can be better understood which helps to identify the features of the query.

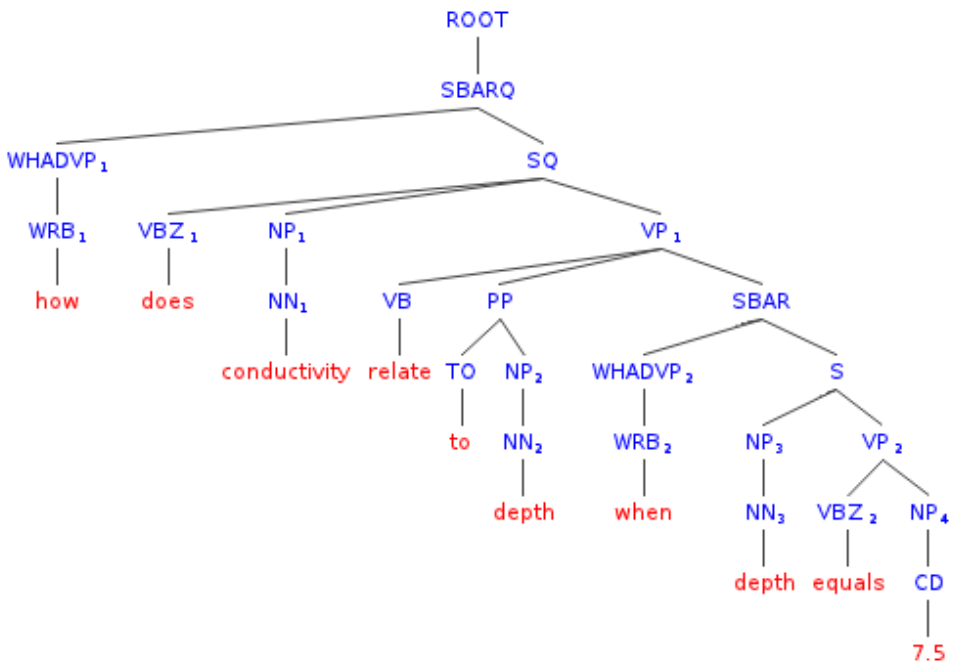

Figure 5. An example of the parse tree for the query "*How does conductivity relate to depth when depth equals 7.5*".

As mentioned early in the section, one piece of essential information in the query sentence is the data of interest. To properly interpret the user's interest in the data, target attributes first need to be recognized. Based on the parsed structures and meta-data characteristics, the algorithm of the attribute recognition can be expressed in four steps:

- Surface match: in this step the original words are examined with regard to the existing attribute names. This step corresponds to the surface part of the meta-data structure. For example, "depth" and "temperature" are the surface words for attributes depth and temperature in a hydrologic dataset, and can be recognized directly from the query "what is the correlation between depth and temperature".
- Meta-word match: in this step the extended terms, i.e. the semantic relations for each attribute name are sought in user's query. For example, the semantic related words for apple include "fruit, produce, food, pome, tree".

- Value match: in this step a collection of values in the data file that belong to each attribute are examined within user's query. Currently, only nominal attributes are considered, where values can be grouped into limited number of categories, such as country, education level.
- Query expansion: in this step I expand certain "open-class words" in the query, more specifically adjective and adverb, by considering its gloss. For example, in the query "*what is the heaviest American car*", the adjective *heavy* refers to the attribute *weight*. By retrieving the gloss of *heavy* from WordNet, "of comparatively great physical weight or density" can be obtained, which links this adjective to the targeted data attribute *weight* properly.

After finding the target attributes, I further investigate if there is any constraint on them. The constraint often appears as a specification on certain values or a value range, like a filter in the database operation. Here I focus on the numerical constraints, since nominal constraints can be recognized in the Value Match step discussed above. Through empirical observation, it is found that several classes of filters can be identified using shallow linguistic characteristics, which include:

- Number appears as an object related to an attribute, such as "*if depth equals 7.5, what is the correlation between conductivity and temperature*". This can be identified by two dependencies in the dependency diagram produced by the language parser: number 7.5 as a direct object of the verb *equals*, and attribute *depth* as a nominal subject of the verb *equals*.
- Number appears as a direct modifier to an attribute, such as "*how many cars have 5 cylinders*". In this case, there is a direct edge labeled "num" (an abbreviation for numeric modifier) between number 5 and attribute *cylinders* in the dependency diagram.

- Number serves in a prepositional modifier to an attribute, such as "*what is the percentage of pH above 7*". This can be handled by identifying the edge labeled prep_above between attribute *pH* and number 7. Similar prepositional modifiers include prep_below, prep_of, prep_than, prep_between, and etc.

These cues can be easily identified in the dependency structures produced by the language parser.

3.2.2.2. Representing Syntactic and Semantic Features

The results from the language parsing step provide complex information about the features of the query. Some of the information is not essential in the procedure of identifying the user's general intention. Identifying the user's intention from the input sentence can be considered a particular case of text categorization, of which the purpose is to automatically classify text or documents into one or more of predefined categories. One of the central issues in text categorization is text representation. This section will discuss the approach for resolving this issue.

One of the common methods for text representation is the bag of words approach. In this approach, general stop words, such as *a*, *an*, or *the*, are removed from the text, and the remaining terms are used as the textural representation. This approach has been widely used in information retrieval primarily because of its simple nature. However, a disadvantage is that each word is considered a possible feature and thus the number of features can be unnecessarily very large. Furthermore, a simple bag-of-words representation does not reflect the word order or sentence structure, for instance, you cannot tell if the phrase "natural language" exists within the text, or appeared as two individual words unrelated to each other. Building upon bag of words approach, some other tactics have been introduced, such as stemming words, noun phrases, and named entities. The concept of named entities recognition is to tag the noun words or phrases into predetermined categories, including date, location, person, organization, and etc. The entity

tagging has the advantage of abstracting the terms in the document and discarding the noisy terms picked up by bag of words or noun phrases.

In *Articulate*, the input texts are sentences that are short in length, and the vocabulary of the corpus is relatively specific. Because of these characteristics, using semantic knowledge could be more beneficial than a general collection of words. Inspired by the named entities approach, I defined six categories of keywords for feature representation: *comparison, relationship, composition, distribution, statistics and manipulation*, to reveal the semantics of user's general intention as described in the design space analysis (Section 3.1.2). The keywords in each dictionary are selected according to empirical knowledge and domain vocabulary. For example, *associate, correlate, link, relate, relevant* are often used in the queries intended for tasks regarding relationship or connection between two or more variables, so they are entered into the relationship dictionary. In addition, the entries in the dictionary are grouped by their lexical category, i.e. noun, verb or adjective. The matching between the stemmed words and the entries in a dictionary is done by a string comparison. Instead of simple word matching, the algorithm first checks the lexical category in the dictionary compared with the part-of-speech tag on the stemmed word, if they are similar then checks whether the two strings are identical. This way certain lexical ambiguity caused by words with multiple function categories can be avoided.

Besides that, the findings from the preliminary user study show that some queries were not correctly answered due to the ambiguity of query's feature. It is possible to improve feature identification by capturing the syntactic characteristics of those queries. Through a close examination, several shallow linguistic features were found that might help the classification of the query:

- Clause Type: clause is a group of words containing a subject and a predicate and forming part of a sentence. A simple sentence consists of just one main clause. A complex sentence contains one or more subordinate clauses. According to the Penn Treebank annotation, the types of clauses include:

- Simple declarative clause, e.g. *I want a pie chart*.
- Direct question introduced by a wh-word or wh-phrase, e.g. *what is the correlation between depth and temperature*.
- Clause introduced by a subordinating conjunction, e.g. *if depth equals 7.5, ….*
- Inverted declarative sentence e.g. *did the weight of cars increase or decrease between 1970s and 1980s*.
- Inverted yes/no question, e.g. *can you apply attribute pH onto color*.
- Verb phrase. e.g. *plot x and y as the x and y axes*.

The type of clauses reflects the overall structure of the sentence, which often varies based on the user's intention. For instance, it is noticed that many queries for the manipulation tasks appear in the form of verb phrase, such as "*Remove mpg from x axis*." "*Add depth as color*." Similarly, a subordinate clause often indicates a specification or filter on the value, for example "*when depth equals 7.5*" in the sentence "*show all the data values when depth equals 7.5*".

- Superlative or comparative adjective or adverb: a query contains a comparative or superlative adjective or adverb often indicates a comparison task or analysis task. For instance, in the query "*what is the heaviest American car*", "*heaviest*" refers to the largest observation of weight; hence this query is a typical statistical request in the analysis task.
- Cardinal number: cardinal numbers often specifies a particular value or value range on an attribute with numerical type. By recognizing this constraint, the data can be better filtered to the user's preference.
- Quantifier: a universal quantifier (e.g. *all, both, every, each)* formalizes the notion that a predicate is true for everything. With regard to this, a query without a specified attribute name but a quantifier, such as "*find the price of all the food between 1989 and 1991*" can be informally read as "*find the price of bread, beef, chicken, eggs,*

apples, bananas, tomatoes, and orange between 1989 and 1991". Besides, a universal quantifier + a pronouns can refer to all the attributes used in the previous query, to illustrate here is an example of queries in discourse:

"*How have the price of beef, chicken and eggs increased over time*"

"*I would like all of them in one graph*"

Based on the above syntactic and semantic feature analysis, a smaller feature space is derived to represent the most important aspects of the query. This feature space is defined as a fourteen-dimensional space. Each dimension describes one feature found in the query, such as the existence of keywords for one class of visual intention, or the number of attribute names appearing in the query. Specifically, the features are:

- comparison_keyword [true/false]
- relationship_keyword [true/false]
- composition_keyword [true/false]
- distribution_keyword [true/false]
- statistics_keyword [true/false]
- manipulation_keyword [true/false]
- timeseries_keyword [true/false]
- visual_primitive_keyword [true/false]
- clause_type [0,1,2,3,4,5]
- superlative [true/false]
- cardinal [true/false]
- quantifier [true/false]
- has_filter[true/false]
- number_of_attributes [0, 1, 2, 3]

The first eight features all have Boolean values: "true" if there is a word in the query matching a keyword in the corresponding dictionary, "false" if not. In particular, the first six categories of features are chosen based on the taxonomy of user's intentions in visualization tasks;

while, the time-series and visual primitive keyword conveys user's direct request on the visual display choice.

The following features describe those shallow syntactic features. The ninth feature indicates the type of clause for the query as labeled at the root of the parse tree, which includes simple declarative clause, direct question introduced by a wh-word or wh-phrase, clause introduced by a subordinating conjunction, inverted declarative sentence, inverted yes/no question or verb phrase. The next four features also have Boolean values. Among them, the superlative feature describes whether a comparative or superlative adjective or adverb is used in the query. The cardinal feature shows whether a cardinal number appears in the query, which is often applicable to queries with data filter. The quantifier feature indicates whether a universal quantifier like *all, both, every, each* exist in the query. And the next feature implies whether a filter is specified. This feature is inferred from the recognition of data constraints discussed in Section 3.2.1.

The last feature can have multiple values: 0 implies no data attribute specified in the query, 1 represents only one attribute, 2 represents two attributes, 3 implies more than two. This feature is also inferred from the recognition of target attributes described in Data Parser (Section 3.2.1).

After defining the feature space, a query sentence can be simply represented as a feature vector. For instance, the query "*How does conductivity relate to depth when depth equals 7.5*" can be expressed as [`f,t,f,f,f,f,f,f,1,f,t,f,t,2`].

3.2.2.3. Classifying the Intended Tasks

As discussed earlier in the design space analysis, identifying users' major intention of the visualization tasks can help translate their imprecise query into precise description, which then guides the Visual Executor to provide appropriate graphical results. The four groups of identified

tasks: sketch, analysis, manipulation and filter, specifically characterize the users' behavior tailored to the natural language environment. For example, sketch tasks are performed when users want to gain an overview of the data. Analysis tasks involve the use of statistical methods such as taking the average or standard deviation on the data. Filter tasks allow users to filter out unintended items. Manipulation tasks describe ways to alter an existing visual representation, such as remapping colors or switching axes.

After defining a smaller feature space, I map the query into a representative feature vector. The feature vector essentially identifies the words that describe the intended visualization. It does not however guarantee that the user is sufficiently precise in their use of their wording. Therefore the feature vector is given to a task classifier that attempts to derive the user's true intent using a supervised learning method. According to the taxonomy of visualization tasks discussed earlier, seven classes are identified: *comparison, relationship, composition, distribution, statistics, manipulation, and filter*. Three widely used supervised learning algorithms were considered for this classification task: Decision Tree, Bayesian Network and Support Vector Machine [Witten11].

The first approach I attempted is based on the decision tree method [Quinlan86]. A decision tree is induced from labeled training datasets. In the tree structure, each internal node involves testing a particular feature. For a feature with Boolean values, the tree will split in two ways depending on which condition holds; for other nominal feature, the number of branches is usually the number of possible values of the feature. A leaf node does not split but gives a resulting class. To classify a new query, upon receiving the feature vector, a route is selected down the tree according to the values of the features tested in successive nodes, and when a leaf is reached the query is classified.

Decision trees are relatively easy to understand, but may become computationally expensive when the number of features increases. A computational-efficient alternative is the Bayesian Network [Pearl85], which represents the probabilistic relationships between a set of

random variables, and allows conditional independence between subset of them. Compared with another popular probabilistic classifier Naïve Bayes, which assumes strong independence between variables, the Bayesian network makes it more general by dealing with datasets that have internal dependencies. The Bayesian network can be represented in a directed acyclic graph (as shown in Figure 6), where each node represents one feature or the class, and each edge represents conditional dependence. For instance, the feature *query_contains_timeseries_keyword* has direct influence on the feature *query_contains_comparison_keyword* so there is an edge between them. Nodes which are not connected represent attributes which are conditionally independent. Each node is associated with a probability distribution that is used to predict the class probabilities for a given instance.

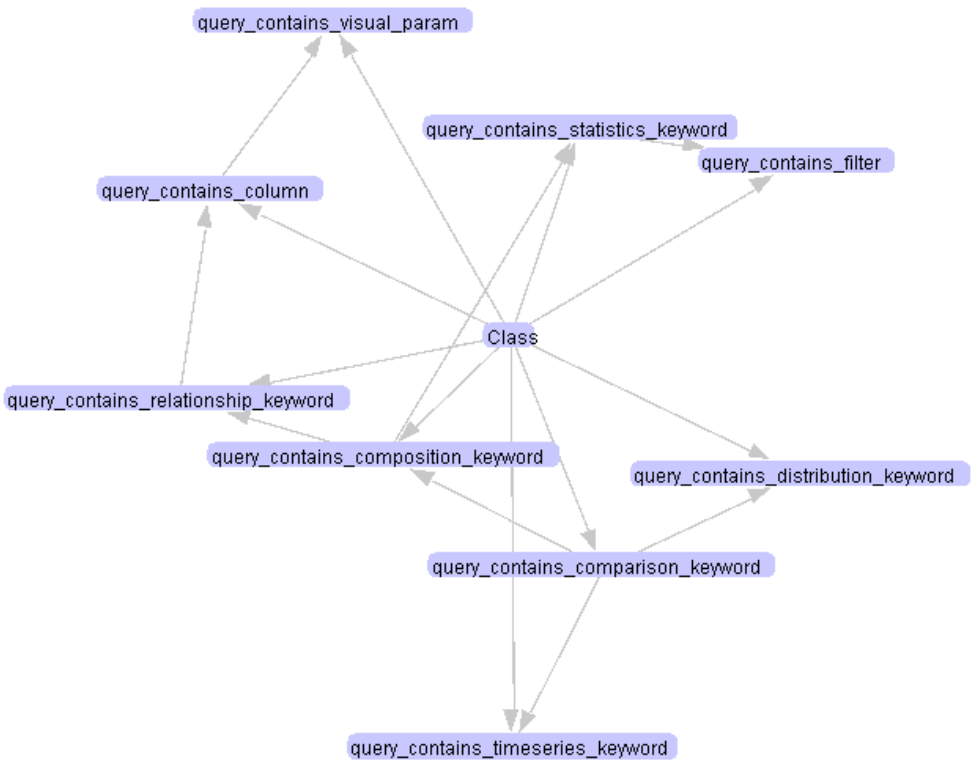

Figure 6. An example of Bayesian Network for classifying users' intentions.

The third classification approach I tried is based on the Support Vector Machine [Cortes95]. It uses a nonlinear mapping to transform the original training data into a higher dimension, from which it can find a linear optimal hyper-plane to separate the data with the maximum margin. A major issue with the support vector machine algorithm is the slow speed for building the model.

In this scenario, each model generated by one machine learning algorithm can be regarded as an expert. It is more reliable to take into account the opinions of several experts rather than relying on only one judgment. Therefore, I combine a decision tree inducer, a Bayesian network learner, and a support vector machine to form a meta-learner, which takes the majority votes from the three basic classifiers. In this framework, the actual classifier was formed by applying this meta-learning method over a corpus of queries collected from a preliminary study. The performance of the meta-classifier will be discussed in Section 4.1.3.

3.2.2.4. Suggesting Visualizations in Context

The visualization task recognized using the meta-classifier gives the user a solution based on the limited training corpus. Hence it may not always be the "best" choice. To help users find the truly intended visualization, a means to allow *Articulate* to suggest possible visualizations was explored. The benefits of providing the suggestions are:

- Help users quickly find their intended visualization, and potentially speed up their data exploration process.
- Offer users more alternative visualizations, which may help them consider alternative perspectives of the data.
- A user's choice in the suggested visualization serves as a context for the automatic decision making in the subsequent tasks.

The algorithm employed to select candidates for suggestion is based on a context-aware meta-classifier. This classifier is similar to the idea used in task classification process discussed in the last section, in which a decision tree inducer, a Bayesian network learner, and a support vector machine are combined together. But different from the task classifier, probabilities are taking instead of majority votes. For each visualization class, the weighted probability is calculated estimates from the three base classifiers, and rank the candidates based on that estimated score. The top three classes in the rank will be presented as suggestions. The measure of the estimated score is defined as follows for each possible class i, i = 1...7, and k = 0, 1, 2, corresponding to three base models: decision tree, Bayesian network and support vector machine:

$$score_i = (1 + c_i)\frac{1}{k}\sum_k p_{ki} w_{ki}$$

p_{ki} - the probability of this instance being classified as class i using model k. Instead of taking equal weights from all these models, the contribution of each model is weighed based on their performance. This performance is measured by w_{ki} – the precision on class i using model k.

Different from a simple classifier, the immediately preceding tasks were taken into account as context for suggestion. Because by achieving the previous visualization goal, users have a high tendency to continue their exploration in a similar way, which means the prior choice of the visualization task will significantly influence user's preference in the subsequent queries. For this reason, I add a contextual factor c_i into the classifier formula, which reflects the use-frequency for class i. The more recently used, the higher this factor is. This contextual factor is formulated as:

$$c_i = \left| \frac{use_i - \mu_i}{\sigma_i} \right|$$

Here μ_i and σ_i are the mean and standard deviation of use-frequency for all the classes. The use-frequency use_i is updated after each query. Notice that the normal score for this factor is taken, so

that more influence is given to the most recently chosen class - the one with a larger distance from the mean. This maintains the coherence of attention in successive tasks.

Maintaining a context of previous tasks not only expedites the realization of users' truly intended visualization, but also plays an important role in resolving references in natural language parsing. For instance, the framework will not be able to classify a single query like "*what about the other variables*", as no direct information about the data of interest or the intended task is given in this sentence. But considering it as an immediate follower of the query "*How has the weight of cars changed over time*", then the referring expression "other" can be linked to the entity under discussion (i.e. "weight"), and therefore the data of interest are the remaining variables. Furthermore, the context set up by the previous query also implies that the intended task should be time-series visualization as well. In this respect, the previous queries and corresponding visualization served as a context within which the new query can be interpreted appropriately.

Figure 7 shows an example of suggestions given by *Articulate*. For the query "*compare the distribution of beef and the distribution of eggs*", the classified major intention is a distribution task, which is shown as histogram in the main visualization. Meanwhile three candidates with high probability or frequent usage, like the comparison (trend) task displayed in time-series chart, are offered as suggestions and ranked in the order of their context-aware classification scores. These suggestions are presented as a gallery of thumbnails on the side of the main visualization. This design enables the user to easily compare alternative visualizations, as well as balances the use of screen real estate. If one of the suggestions is preferred over the default visual result, the user can simply double click on the choice to bring it to the main visualization panel. This selection, as an update in the context, will immediately influence the decision making in the following tasks. The user study reveals that providing context-aware suggestions greatly benefit novice users in their data exploration.

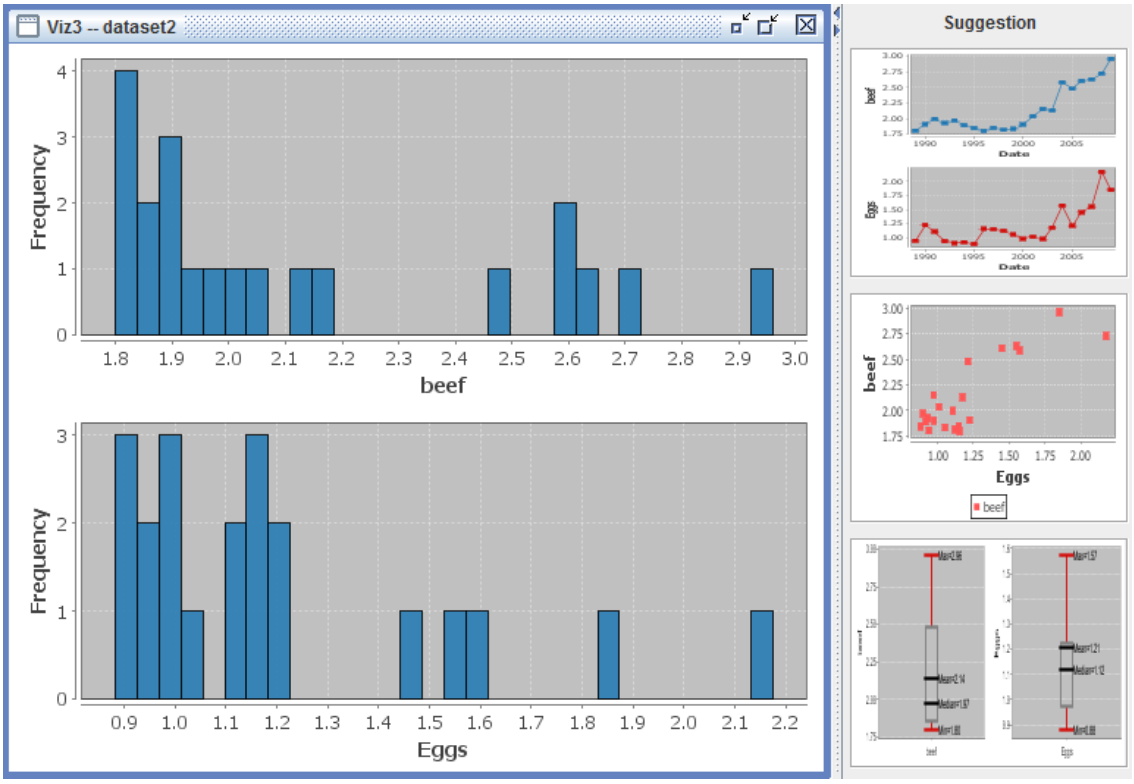

Figure 7. An example of suggestion results for the query "*compare the distribution of beef and the distribution of eggs*".

3.2.2.5. Generating SimVL Commands

The information generated from the previous steps, especially the classification results and the data of interest, will be passed into the Visualization Executer to guide it through the graph generation. These intermediate specifications must be expressive to describe the semantics of user's various intentions; meanwhile, they must also be consistent and easy to translate into low-level instructions for graphics engines. I propose a Simplified Visualization Language (SimVL) to serve as the medium. SimVL is specified in a formal grammar, in which each command expresses a specific assignment. A command is typically composed of two parts: an action that describes the general purpose of the task, and a list of parameters that explain the

setting details. Four major types of SimVL commands are defined to accommodate the different purposes of visualization tasks:

- Sketch commands are the ones that describe the semantics of general visualization task. The grammar of this command is presented below:

  <sketch> := PLOT <class_of_task>

  | PLOT <class_of_task> OVERTIME

  As shown above, a statement is composed of action Plot and one or two parameters. The first parameter indicates the classified sketch task, including RELATIONSHIP and COMPARISON. The second parameter indicates whether time-series data is required.

- Analysis commands are normally used when the user is interested in the statistical feature of data. For example, minimum, maximum, average, distribution, composition etc. So the commands are defined as action ANALYSE followed by two parameters: one indicates the feature, which can be MIN, MAX, MEAN, MEDIAN, RANGE, DISTRIBUTION, COMPOSITION; the other lists the target attributes. And optionally there is another parameter of attribute indicating grouping operations:

  <analysis>:= ANALYSE <feature> OF <attribute_list>

  | ANALYSE <feature> OF <attribute_list> GROUPBY <attribute>

- Manipulation commands are used to alter existing visual primitives. As Cleveland et al. [Cleveland84] have identified, there exist a set of basic means to encode data on graphs, such as position, length, direction, area, curvature, shading and etc. The manipulation commands are defined to express these graphical encodings, for example:

  <manipulation> := SETX <attribute_list>

  | SETY <attribute_list>

  | SET AREA AS <attribute>

| SET COLOR AS <attribute>

Similar to the sketch commands, they are also defined as a list of statements. However, these commands do not enforce the generation of a new graph, but focus on the mapping or assignment of visual metaphors.

- Filter commands are mainly used to select the desired pieces of data, such as "*temperature below zero*", "*years between 1950 and 1990*". One way to specify such selections is to use a generalized constraint expression of the form X R, where X is the constrained variable, R is the constraining relation. Therefore, the above examples can be described as "temperature LT 0", "year BT 1950, 1990", where LT stands for less than, and BT means between. And the grammar of filter commands can be expressed as:

  <filter> := Filter <X> <R>

The purpose of using SimVL is to provide a standard and simplified format for user's request, in order to facilitate the visualization execution.

3.2.3. Visualization Executer

The Visualization Executer reads the SimVL commands, and the specification of various types of visualizations, and uses a heuristic algorithm to pick the most appropriate graph. Just as a visualization expert might weigh the pros and cons of different graphs in the determination of a desired plot type, the Executer works as a smart agent performing a similar reasoning process.

3.2.3.1. Reasoning Algorithm

As discussed in the visual display level design analysis, there are a couple of important factors in the visual execution, including the property of the data (such as the number of variables, data types, and whether the variables change over time), the effectiveness of different visual

primitives. Taking into account these factors, the rules for generating a graph from various SimVL commands are derived.

For sketch commands, the choice of a specific visualization is contingent upon factors like the number of variables, types of data, importance of attributes, and effectiveness of visual primitives. Figure 8 summarizes the reasoning algorithm for sketch commands.

To illustrate how it works, here is an example. Given a 1983 ASA Data Expo dataset on automobiles, a query "*how has MPG changed since 1970*" will be translated into a sequence of SimVL commands:

FILTER *Year* GE *1970*
PLOT COMPARISON OVERTIME
SETY *MPG*

The first command specifies the selection of data that is defined as a constraint over the attribute *year*, of which the value must be greater than or equal to 1970. The next command indicates the query is a comparison task. As discussed earlier, for quantitative comparison, using position or length to represent the value is perceptually much more effective than other visual primitives, so a scatter plot, line chart, or bar chart are most desired. Then, it follows the comparison sub-algorithm to make the chart. The first decision to make is whether the independent variable is ordinal. Since there is an OVERTIME parameter in the command, which indicates the independent variable is *year*, the answer for this test is "Yes". For an ordinal variable, particularly time-series attribute, connecting the series of points with line segments could reveal the trend in data clearly; thus the line chart becomes the graph of choice. Next step is to look into the dataset to find whether there is a unique *MPG* value on each year. Unfortunately it is not the case. So an aggregation operation is applied to the original dataset to get a summary for the yearly amount, including a mean value and data range, which are then used to populate the final graph as a line chart with whisker (as shown in Figure 9). Similar to the box-and-whisker plot for statistical

analysis command, the top and bottom ends of whiskers indicate the maximum and minimum *MPG* values for each year.

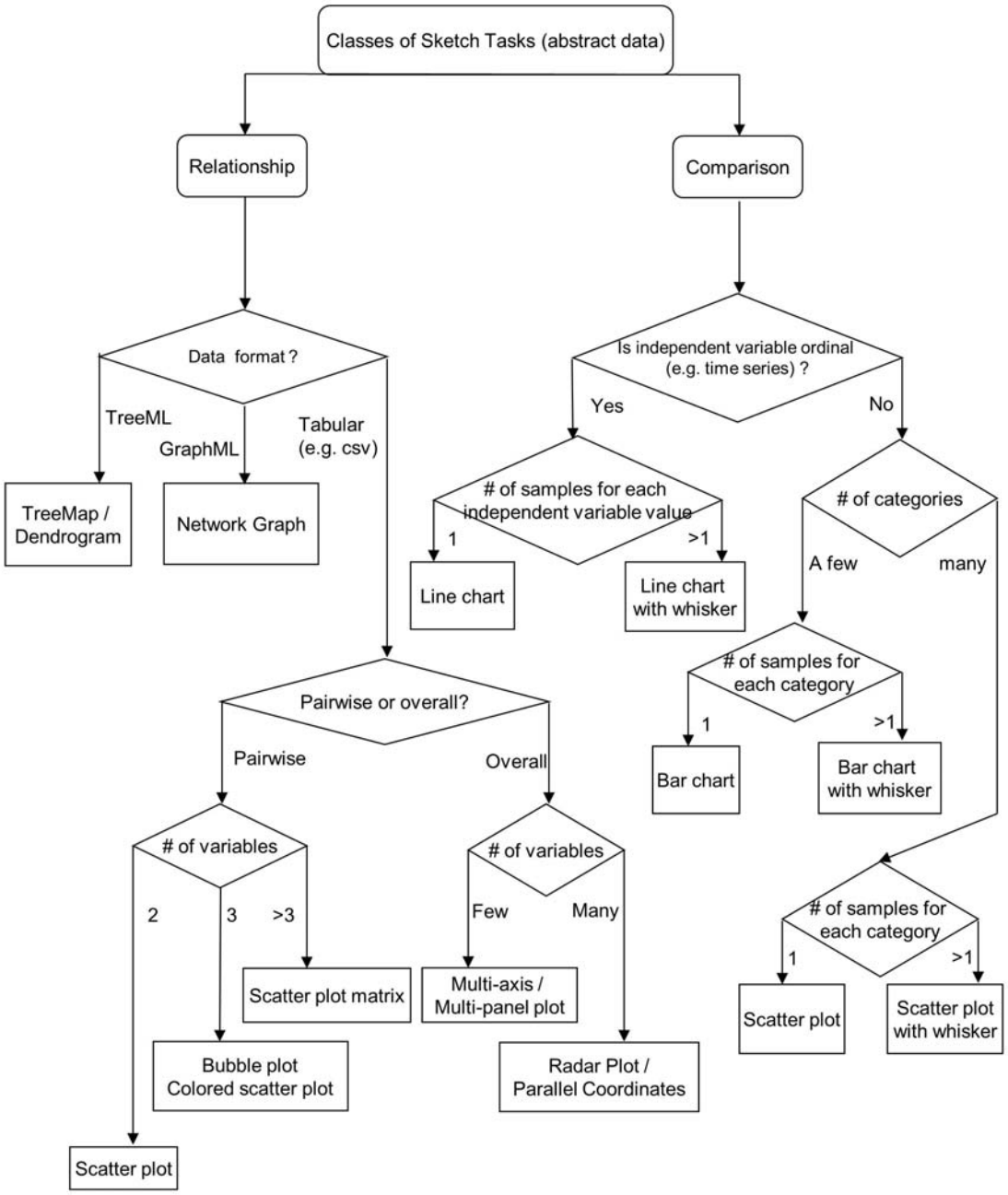

Figure 8. The graph generation algorithm for sketch commands.

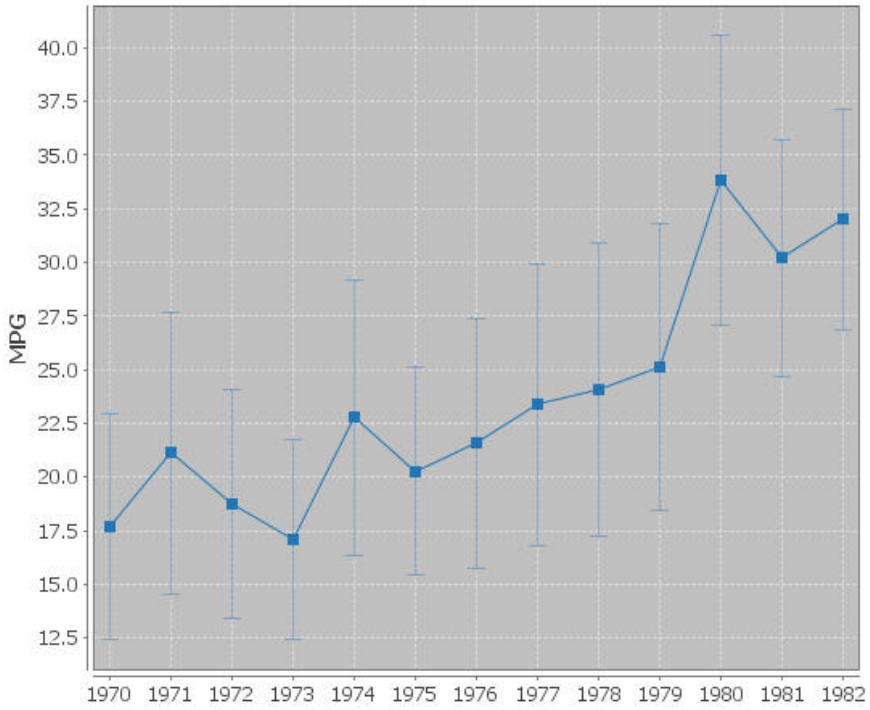

Figure 9. Result for sketch and filter commands translated from "*how has MPG changed since 1970*" with regard to a 1983 ASA Data Expo dataset on automobiles.

The reasoning algorithm for analysis commands is shown in Figure 10. For those tasks focus on the basic statistical features of data (such as minimum, maximum, average, and quantile), a box-and-whisker chart is a convenient way of graphically depicting these features: the ends of the whisker represent the minimum and maximum of the data, the bottom and top of the box are always the 25th and 75th percentile; while, the line in the center of the box can be mean or median based on user's request. The spaces between the different parts of the box indicate the dispersion in the data. Figure 11 gives an example.

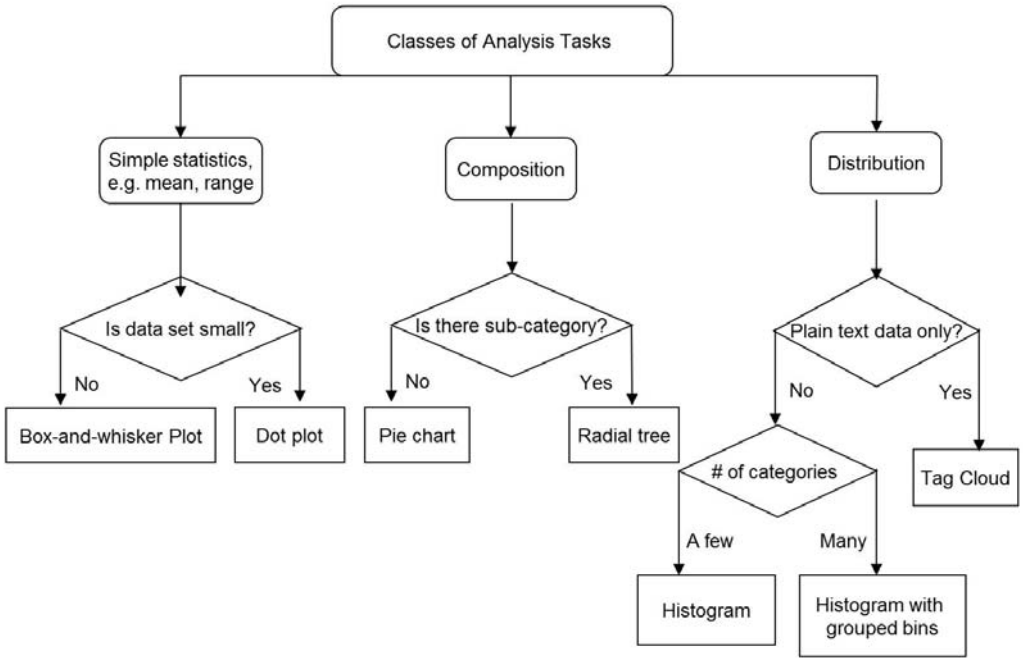

Figure 10. The graph generation algorithm of for analysis commands.

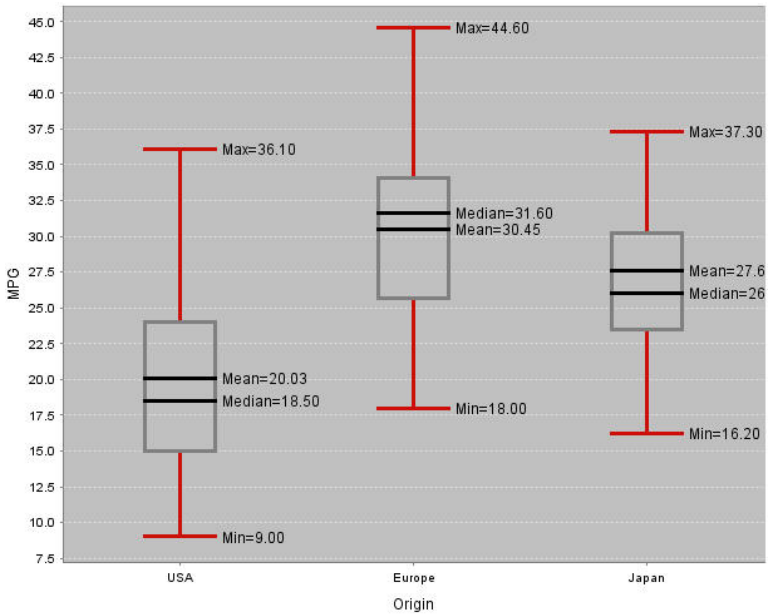

Figure 11. Result for analysis commands translated from "*what is the average MPG by country of origin*" with regard to the same dataset as Figure 9 (a 1983 ASA Data Expo dataset on automobiles).

On the other hand, manipulation commands are typically adjustments made to an existing visualization, for example switching axes, choosing to color data points based on the values of another attribute (as illustrated in Figure 12). Hence, the Executer only needs to recognize the visual primitives from the SimVL command, and map the specified attribute onto that. And filter commands are typically attached to sketch commands or analysis commands with constraints on data values, so the Executer's job is to understand the constraints and filter out unintended data before plotting it.

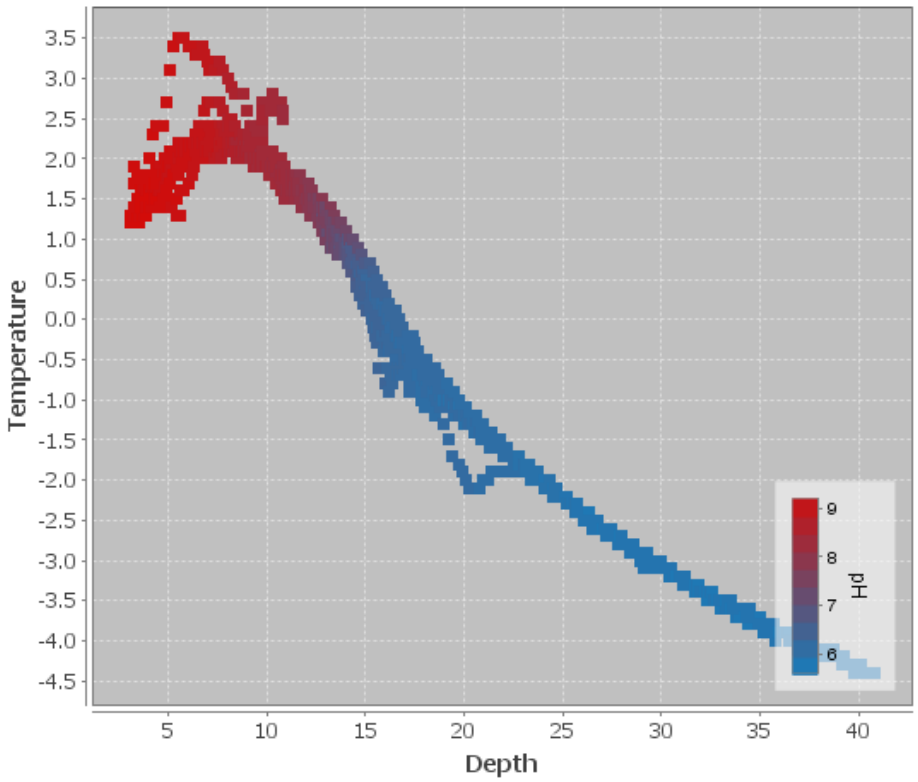

Figure 12. Result for manipulation commands translated from "*can you color the points by pH*" following a query "*what is the correlation between depth and temperature*" with regard to a 10-attribute hydrological dataset.

3.2.3.2. Data Transformation

As mentioned above, the form of data, whether it meets the requirement of a certain graph type, can affect the efficacy of the resulting visualization. Some graph types best visualize fewer values, whereas others can encode an arbitrary number of values. To take advantage of this feature and maximize each graph's capability of sense-making, a data transformation is necessary before the visualization is actually performed. The following transformations are included in *Articulate* for this purpose:

- Counting: refers to the counting of distinct values in a data attribute. If an attribute has only a few distinct values, when applied to a histogram or pie chart, the difference between each segment could be easily perceived. But if the number of distinct values is too large, further transformation might be needed.
- Partitioning: is used to discrete a continuous attribute. Particularly for graphs, such as histogram, defining a limited number of regular-sized bins, and then partitioning the data based on the bins, will make the distribution pattern much more salient.
- Grouping: is similar to partitioning, but is constructed on the basis of the semantics in a user's query. For instance, a query "*what is the average MPG by country of origin*" (Figure 11) indicates a grouping operation that collects all the data entries that contain *MPG* values and groups them by country.
- Summarizing: includes basic aggregation operations, such as average, minimum, and maximum, that can be applied to a single attribute or grouped values. Summarizing applies a many-to-one transformation to convert a set of samples into a couple of representative units. For instance, in Figure 9, the summary of *MPG* for each year, is calculated and applied to a line chart, where average values serve as the connecting points of the line, minimum and maximum make up the range whiskers.

- Sorting: refers to arranging the values in a certain sequence, such as chronological order or numerical order. After applying sorting operation, the minimum, maximum and relative orders among elements becomes much clear, which could potentially reveals more patterns of the data.

Employing data transformation before plotting them helps to make better sense of the data visualizations. As the users explore the data, partitioning or grouping helps discover major difference between pieces of data, and summarizing can hide uninteresting data and reduce the size of a large dataset.

4. IMPLEMENTATION AND INTRINSIC EVALUATION

Given the design analysis for the automatic generation of visualizations through the use of natural language as a primary interface, the question remains of how to actually implement each component in the framework. This chapter presents the implementation details. The mechanism of a multi-modal interface is described first, followed by the parser for natural language interpretation. Next, the performance of different classifiers is examined. Then the functions of different graph engine employed in the Visualization Executer are introduced. Finally, I discussed the advantage of integrating such system into a high resolution display environment.

4.1. System Implementation

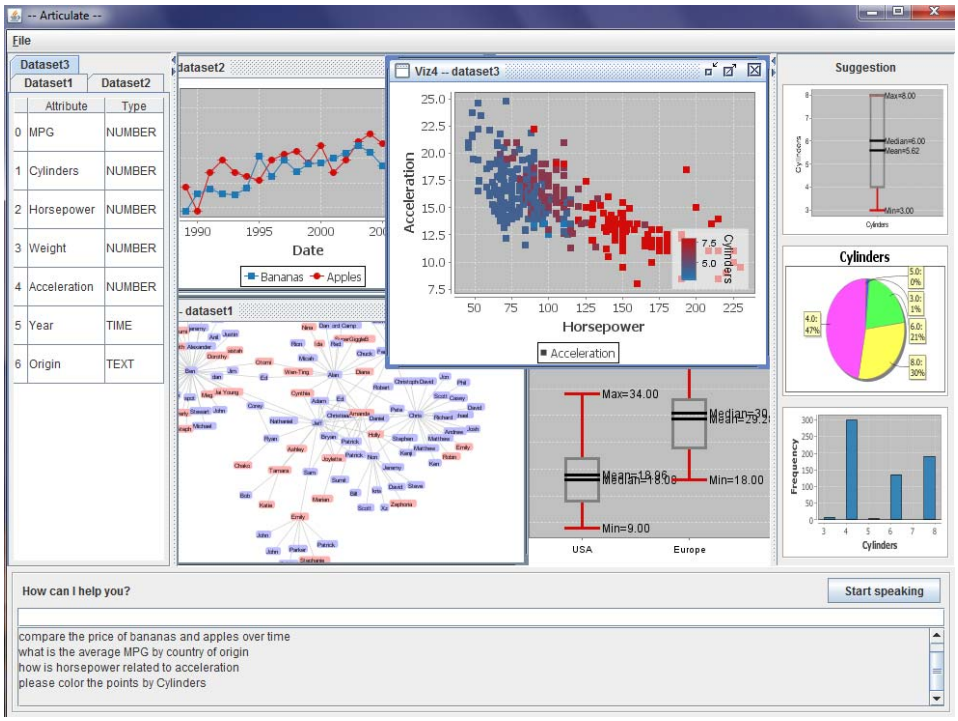

Figure 13. The user interface for the *Articulate* system.

The *Articulate* system is developed in Java. Figure 13 shows a screenshot of the prototype. A user speaks into the system without knowledge of visualization commands and the translated text is shown on the bottom-most window. The resulting visualization is shown in the main view above it. Both the suggestion panel (on the right side of the main visualization) and data window (on the left side) are collapsible to allow users to focus on the intended visual result. In the input panel at the bottom, there is a query history window which maintains all the previous inquiries allowing users to verify if the system has correctly interpreted their natural language input.

4.1.1. Multi-modal Interfaces

Even though the common mouse and keyboard combination is the primary input device for most existing visualization systems, it is inefficient for a natural interactive environment. As in such scenario, an easy and less disruptive input method is needed while users are carrying out their tasks. Therefore, I integrate a speech interface into the system.

In *Articulate*, users can input their queries by speech or text. To recognize the speech input I start by using the Chant Speech Kit. Based on Microsoft Speech API, this kit translates a spoken query into a plain-text sentence. But since that is aimed at helping developers building speech applications rapidly, the kit lacks flexible control to the acoustic and language model. Additionally it is based on an older version of Microsoft Speech API, while the newer version is not released as a freely-redistributable component but integrated into the operating system only. So during the iterative design I changed the speech engine to an open source alternative – Sphinx toolkit, a leading speech recognition toolkit developed at Carnegie Mellon University with various packages used to build speech applications in Java [CMUSphinx]. Using Sphinx toolkit, I can build my own statistical language model from a small domain-specific corpus, and create the

corresponding dictionary which maps words to their phonemes. Since the purpose is to build a speaker-independent interface, the acoustic model I chose, which describes sounds of the language, is the standard Wall Street Journal acoustic models. The user study showed that language models built in this way was functional for limited language and control tasks.

As explored in the taxonomy of different visualization tasks, there are certain tasks which can be completed easily using mouse drags, but difficult to articulate aloud. Therefore, in the interface implementation, traditional mouse interaction is also supported, such as zooming and brushing, tooltips on pointing, as a complement to natural language interface for supporting those types of tasks that are easy to perform with a GUI but cumbersome to speak.

To evaluate the accuracy of the speech recognition in *Articulate*, I recorded the sentence-wise recognition error in my user study. The recognition error rate turned out to be 30% for all subjects, and 40% for native speakers. The main reason for this low accuracy was that the speech recognition system was not trained for each individual user. And users were allowed to speak in a natural rhythm rather than required to make a deliberate pause between each word. To minimize the error caused by the speech recognition, one possible solution is to train the system for the particular user in a specific domain application. Another approach could be investigating the use of gesture input as a complementary interface to voice input, because often time verbal communication can be enhanced or further clarified through a variety of different gestures.

4.1.2. Language Parser

One of the major components in the framework of *Articulate* is the Input Translator, which takes a plain-text input and interprets its intention as a precise request for a specific visualization task. The Stanford Parser is leveraged for this interpretation. The Stanford Parser is a package of Java implementation of probabilistic natural language parsers, which can output

various analysis formats, including part-of-speech tagged text, phrase structure trees, and a grammatical relations (typed dependency) format. For instance, the sentence

How has the price of apple changed over the years

can be automatically parsed into a phrase structure tree and a dependency diagram (as shown in Figure 14).

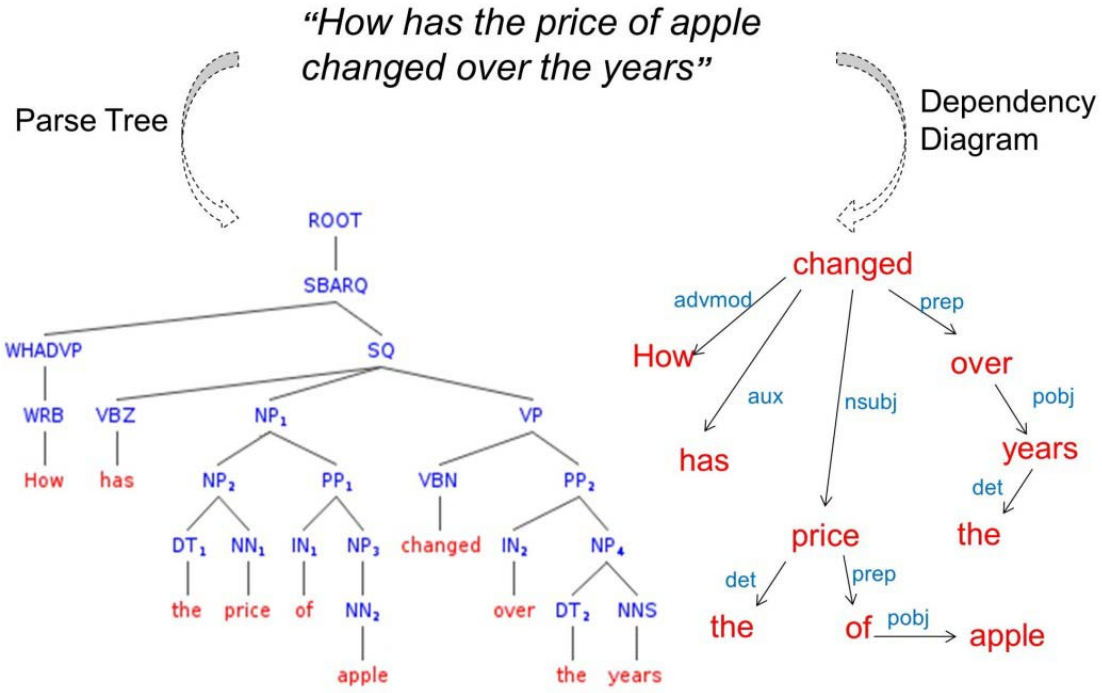

Figure 14. An example of syntax tree and dependency diagram parsed from a query "*How has the price of apple changed over the years*".

Another popular natural language parser is the Berkeley Parser [Petrov06]. It is also based on Probabilistic Context-Free Grammar (PCFG) to analyze the grammatical structure of natural language and assign the most likely parse tree. By comparing each pair of the parse trees generated from both the Stanford Parser and the Berkeley Parser for all the queries collected in

the preliminary study, I noticed the Berkeley Parser produced slightly more errors. And most of the errors happened in the following scenarios:

- There is a domain specific term in the query. For example, given the request "*can you apply the attribute pH onto color?*" for a hydrologic dataset where pH is an attribute for the chemical measurement. The Stanford Parser can correctly group the words "the, attribute, pH" together as a noun phrase, while the Berkeley Parser will recognize pH as an adjective by mistake, and form an adjective phrase of "pH, onto, color" (As shown in Figure 15). The reason for this error could be that the word pH is not a general term; hence the Berkeley Parser is not able to correctly identify its lexical category and functions in the sentence.

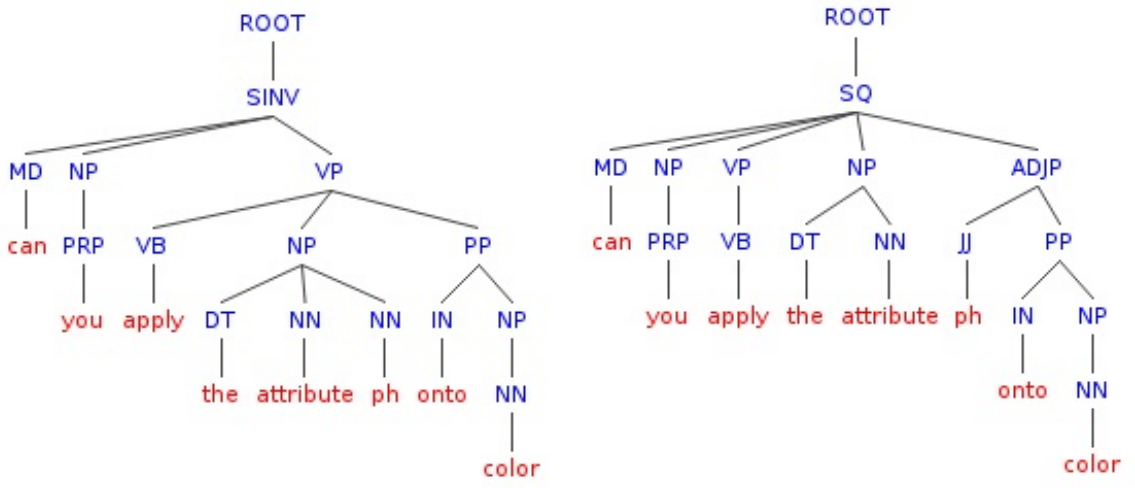

Figure 15. Syntax trees parsed from a query "*can you apply the attribute pH onto color*" by the Stanford Parser (left) and the Berkeley Parser.

- There is a complex modifier for the targeted attribute. It usually happens when the user specifies a numerical constraint. For instance, in the query "*plot the cars that have over 20 mpg and 6 cylinders*". 20 is a direct modifier to mpg and 6 is a direct

modifier to cylinders. These constraints can be easily identified from the parsed structure given by the Stanford Parser (as shown in Figure 16). However, the Berkeley Parser erroneously recognized "over 20 mpg and 6" as a complex quantifier phrase for cylinders, which will result to incorrect identification of value constraints on the attributes mpg and cylinders, and hence produce inappropriate visual results to the user.

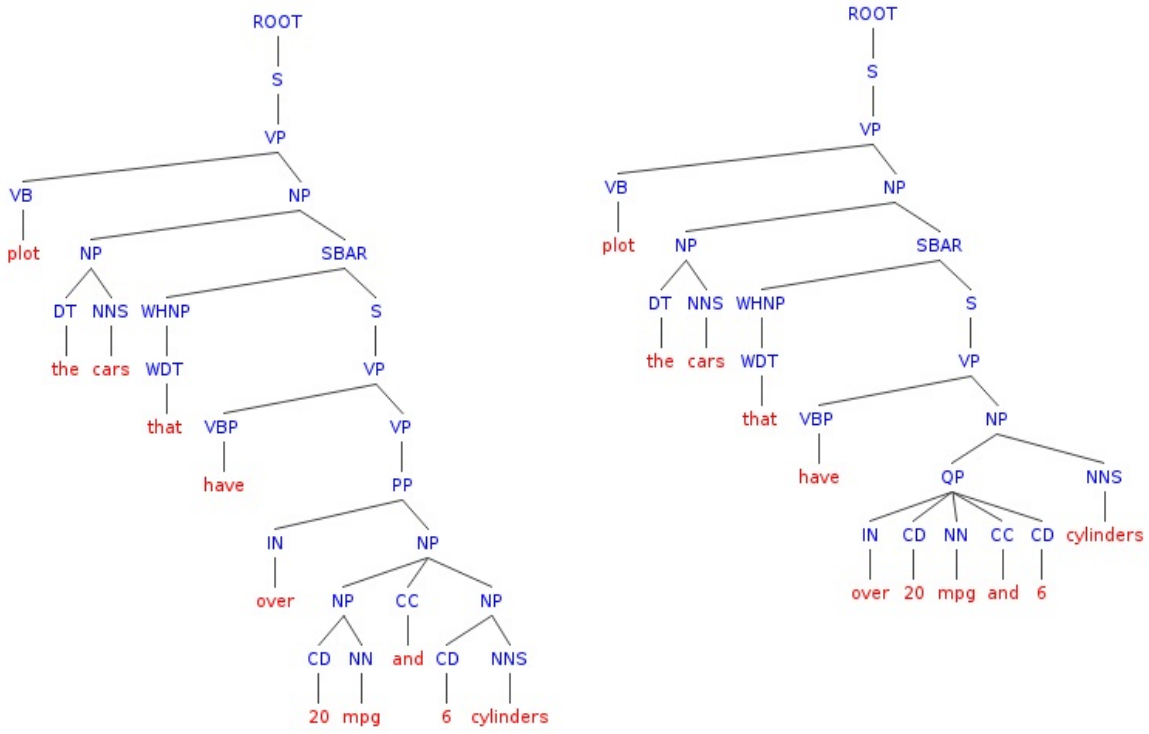

Figure 16. Syntax trees parsed from a query "*plot the cars that have over 20 mpg and 6 cylinders*" by the Stanford Parser (left) and the Berkeley Parser.

Based on the above observations and analysis, I chose to use the Stanford Parser in *Articulate*. However, when I attempt to extract the time-series feature from the user's input, it is noticed that some temporal phrases are not correctly recognized by the Stanford Parser, especially

prepositional phrase or adverb phrase such as “from 1990 to 2005” or “since last year”, which are crucial in the understanding of temporal restrictions in a user’s query. To fix this problem, I integrate ANNIE [Cunningham02], an information extraction system distributed with GATE (General Architecture for Text Engineering, a computer architecture for a broad range of natural language processing tasks) to my language parsing process. ANNIE takes the text input and recognizes named entities in several default categories, such as people, location and date. In *Articulate* only the annotation for date is needed. I tested ANNIE using the queries collected in the preliminary study, and was pleased to find it identified the temporal entities appropriately.

4.1.3. Meta-learning Classifier

To classify the intended tasks, a decision tree inducer, a Bayesian network learner, and a support vector machine are combined to form a meta-learner, which takes the majority votes from the three basic classifiers (as discussed in Section 3.2.2). In the implementation, the actual classifier was formed by applying this meta-learning method over a corpus of 122 queries collected from a preliminary study using WEKA API [Hall09] (an open source collection of machine learning algorithms for data mining tasks). Specifically, the decision tree inducer is implemented by the J4.8 algorithm, which is a later and slightly improved version of the popular decision tree learner C4.5 [Quinlan93]. A hill-climbing method is applied to search over the possible Bayesian network space and find the optimal network structure. And the sequential minimal optimization algorithm is employed to train a support vector classifier.

In an effort to quantify the performance of my meta-learning classification method and understand the variance in performance with the three basic supervised learning algorithms, two metrics are chosen for evaluation: accuracy and precision. Accuracy is measured as the percentage of correctly classified instances, for example in the test, 19 queries are labeled as comparison tasks, but only 17 of them are correctly classified using the decision tree learner, so

this decision tree classifier's accuracy for the comparison task is 89.3%. Then I take the average of the accuracy for each category of tasks, and obtain the average accuracy for this classifier. The other measurement precision is calculated as the fraction of instances classified as each class that actually belong to it. For example, there are 51 instances classified as relationship tasks using the decision tree classifier, of which only 47 are truly relationship tasks, so the precision of this classifier on relationship task is 92.2%. High precision indicates that the algorithm returned more relevant results than irrelevant ones.

To perform the test, I use a 10-fold cross-validation method over the dataset collected from the preliminary study. In the 10-fold cross-validation strategy, data is divided randomly into ten approximately equal sized parts, in which various classes are represented in approximately the same proportion. Each round, one part is held up for test and the remaining nine-tenths are used for training. Repeatedly, the whole procedure is executed a total of ten times on different training sets. Finally, the ten performance estimates are averaged to yield an overall performance measure. All the classification models were built based on WEKA API. The obtained performance results can be found in Table I. As shown in the table, the meta-learner works best in terms of accuracy and precision. Hence it is chosen for the intended task classification.

Table I. EVALUATION RESULTS FOR DIFFERENT CLASSIFICATION METHODS

Classifier		Accuracy	Precision
J48 Decision Tree	14-dim[a]	89.3%	90.9%
	10-dim[b]	89.3%	91.5%
Bayesian Network	14-dim	88.5%	90.6%
	10-dim	89.3%	91.5%
Support Vector Machine	14-dim	86.1%	86.2%
	10-dim	86.9%	86.9%
Meta-Learner	14-dim	91.0%	92.7%
	10-dim	91.8%	93.0%

[a] Query data are labeled in the fourteen-dimensional feature space
[b] Query data are labeled in the reduced ten-dimensional feature space.

To classify users' different intentions appropriately, it is necessary to extract relevant information from the input sentence and label them in a consistent form. Hence a fourteen-dimensional feature space is defined, and each query is represented as a feature vector (as discussed in Section 3.2.2.2). This high-dimensional space covers most characteristics of the input data. But an outstanding question remains: whether they are the most relevant features for the data. When the dimensionality increases, the volume of the space increases so fast that the available data may become sparse for reliable statistical analysis. To avoid the effect of such curse of dimensionality, and speed up the learning process, feature reduction is needed to eliminate those features which may be redundant or contribute little to the decision making.

Feature reduction algorithms are typically divided into two categories: subset selection and feature ranking. Subset selection algorithm often involves a search algorithm that searches through the space of possible subset of features and evaluates each subset for the optimal subset. While feature ranking algorithm ranks the features by a metric and eliminates all features that do

not achieve an adequate score. To identify the most significant features in the multi-dimensional space, both algorithms are applied to the data collected from the preliminary study using WEKA. In the feature ranking algorithm, information gain is used as the attribute evaluator and all the attributes are ranked based on their contribution: the top one is relationship_keyword with over 0.75 score; while, most of the attribute has an information gain more than 0.24, there are four attributes: cardinal, manipulation_keyword, superlative, and quantifier, which have less contribution (with information gain all below 0.15), so these four attributes will be discarded. On the other hand, I applied the greedy hill climbing search algorithm, a popular subset selection method, to the same data, which automatically picks ten best attributes. It turns out to be the same subset as selected using the feature ranking algorithm. To evaluate the accuracy of the reduced feature space, it is compared against the full fourteen-dimensional feature space using cross-validation over various classifiers. Table I shows that the reduced feature space has even better performance than the full fourteen-dimensional feature space. By reducing the dimensionality, it not only speeds up the learning process, but also enhances generalization and human-interpretability of the model.

4.1.4. Graph Engines

The final phase in the framework is the generation of the graph, which is completed in the Visualization Executer. The Executer reads the SimVL command, and the specification of various types of visualizations, and uses a heuristic algorithm to pick the most appropriate graph. To understand the sequence of SimVL commands, I designed a SimVL interpreter. The interpreting process involves three basic steps:

- Parse each command line into tokens, and put them onto token stack;
- Reduce token stack into action list;
- Execute the action list.

Given this interpreter, the grammar of the SimVL commands can be extended or revised easily.

The precise action for the graph generation follows the reasoning algorithm (as discussed in Section 3.2.3) and provides a standard interface for various graph engines. In this initial work I limit the types of visualizations that can be generated to standard 2D graphs and plots, as well as basic 3D scalar visualizations. Therefore JFreeChart, Prefuse and VTK engines are employed to perform the graph generation. JFreeChart [JFreechart] is a free chart library for Java, which supports a great number of standard 2D plots. The ones embedded in *Articulate* are: line chart, bar chart, scatter plot (matrix), bubble plot and radar plot for the sketch tasks, and pie chart, histogram and box-and-whisker plot for the analysis tasks (examples are shown in Figure 17). Prefuse [Heer05] is a toolkit for creating interactive data visualizations, which specializes in visualizing data of graph and tree structure, and producing hierarchical graph like treemap, network diagram and tag clouds. VTK [VTK] is an open-source software system for 3D graphics and visualization that supports a wide variety of visualization algorithms including: scalar, vector, tensor, texture, and volumetric methods. In the current prototype, only scalar data is visualized using 3D glyphing techniques. However, in the future it is fully intended to extend this to accommodate complex visual representations such as those commonly used in scientific visualization (for instance volumetric or streamline visualizations).

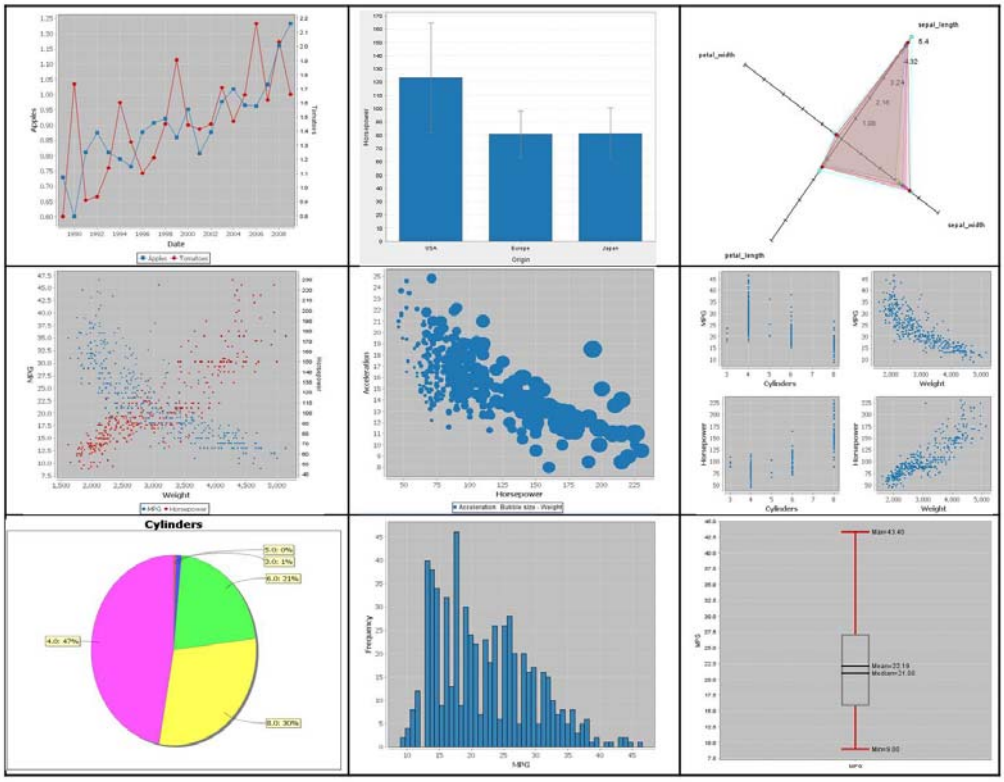

Figure 17. A collection of chart samples for various users' queries generated by JFreeChart.

4.2. Scalable Display Environment Deployment

The *Articulate* system has been deployed on our 25-foot Cyber Commons wall (as shown in Figure 18). The display wall's high resolution and large screen characteristics make it more apt to be used as a "collaboratorium" where people can congregate in front of the display. In such an environment using the traditional mouse and keyboard interface to make a query or take notes requires an elaborate sequence of button taps and keyboard entries, which can potentially become tedious. Instead, natural language via speech offers a more direct and natural means for interaction. It makes communication with the display wall more efficient without increasing the initial learning curve. And speech can be used at a distance, which makes it ideal for hands-busy and eyes-busy situations. It is omnidirectional and so can communicate with multiple users. The

approach de-emphasizes the physical manipulations needed to create a desired graph, so that the user can focus on asking questions about the data. Integrating *Articulate* into this environment allows speech input as an additional modality for interaction, simplify the creation of effective visualizations, and therefore facilitating group-oriented data exploration and problem solving. Meanwhile, the large display space provides an expansive canvas for presenting and comparing multiple visualizations at the same time.

Figure 18. Using *Articulate* on the 25-foot Cyber Commons wall in the Electronic Visualization Lab at UIC.

5. USER STUDY AND EVALUATION

The framework of *Articulate* proposed in the previous chapters has a number of theoretical advantages. However, many theoretical frameworks have been proposed that do not find eventual implementation. The success of the approach will ultimately depend on whether the users perceive improvements in their experience. This chapter describes the design of user studies to validate the approach of *Articulate*.

Based on the iterative design of the framework, two user studies were performed at different stages of the design. First, a preliminary study was carried out to evaluate the approach and explore the limitations. Then a case study was presented in the visualization for domain science, which demonstrated how to expand the framework to domain-specific problems. Finally, a formal user study was conducted. The details about the study methodology, including purpose of the study, recruited subject, tasks, data collection procedure were given, followed by quantitative analysis and observations.

.

5.1. Preliminary Study

5.1.1. Study Methodology

I conducted a preliminary study to evaluate the usability of the proposed approach. The study consisted of a comparison between users of *Articulate* versus a popular graphing tool such as Microsoft Excel. The goal for this exploratory study was to:

- Test the usability of the approach,
- Explore limitations of the prototype system,
- Collect natural language data for further analysis.

Participants

Eleven subjects were recruited for the study (two females and nine males), all of whom are computer science students and have some experience with visualization tools. All the subjects have used Excel before and are familiar with its basic capabilities. Two of them were familiar with Excel's graphing features, eight reported low usage of these features, and one had no experience. During the study each participant was asked to work on the tasks individually. The main reason for recruiting these subjects is that they all have some knowledge about visualization techniques, and have used software to obtain information from data by the means of graphics instead of just numbers. However, they had very little experience with natural language interfaces, especially using such kind of interface to guide visualization tools. This was desirable since it allowed me to investigate novel behaviors that emerged during the study, as opposed to behaviors based on preconceived ideas and experiences.

Procedure

The participants did the study in our lab using a PC running Windows XP operating system. The study consisted of two sessions using two different software systems respectively: *Articulate* and Microsoft Excel. The version of Excel they used was Microsoft Office Excel 2003. And the testing version of *Articulate* was a prototype without suggestion capability. In the first session, the participants were expected to use *Articulate* to perform some data analysis tasks. Each of them was presented with one or two of the following datasets: a hydrologic dataset which contains 10 attributes and 4000 instances; a 1983 ASA Data Expo dataset on automobiles, with 8 attributes for 406 different cars; and Average U.S. Food Prices dataset which contains prices for eight different kind of food over 11 years, published by U.S. Bureau of Labor Statistics. The participants were given 20 minutes to make different queries to *Articulate* to perform three successive tasks:

- Find meaningful correlations among the data attributes.

- Find as many trends as possible about the data.
- Find other interesting features about the data.

The tasks were chosen based on the analysis of different categories of user's intentions for visualization tasks. All tasks were open-ended informational tasks to encourage a full, meaningful answer using the subject's own knowledge and feelings. Participants were briefly introduced to the *Articulate* system ahead of the session. During the session, the subjects were expected to think aloud. After each query, they were asked to identify their purpose of each query based on six categories: comparison, relationship, composition, distribution, statistics, and manipulation. In the second session, the same tasks were repeated using Microsoft Excel. This time each of the participants were given six queries, which were picked from the correctly classified queries in the first session. The subjects were then asked to plot the visual results using Excel. After that the subjects were interviewed for their opinions on the resulting Excel charts.

Data Collection

All the queries the users asked to *Articulate* were logged during the session with time stamps. Each resulting chart displayed on the screen and the users' voice during the tasks were also recorded. In the second part of the study, where Microsoft Excel was employed, snapshots were taken to log each chart plotted using Excel's graphing feature. After both sessions, I summarized the number of graphs produced, their types, and the duration needed to create a graph that resulted in a discovery. Following the completion of the study, a brief interview was carried out. The Interview focused on participants overall impression as well as the ease of use of the *Articulate* system compared with Microsoft Excel, and any suggestions the participants had for improvement.

5.1.2. Results and Findings

In the study, all of the subjects' queries are logged in sequence. Each query is labeled with its actual intention identified by the subject. The intentions are divided into seven categories: comparison, relationship, composition, distribution, statistics, manipulation and other. Subjects' choice and the resulting classification generated by the system are distinguished. A query is counted as incorrectly classified if its system classified category is different from the subject's label. The result is shown in Table II, where the queries for comparison and relationship tasks are merged into the sketch category, and composition, distribution and statistics tasks are merged into the analysis category.

All subjects in the first part of the study were able to use the current *Articulate* system to find some data features and trends. But as shown in the table, for each category of query there is an average of 15% classification error (SD=0.04). To discover the cause of this, I looked through each incorrectly classified query, and made a couple of interesting findings:

Table II. CLASSIFICATION RESULTS FOR EACH CATEGORY OF VISUALIZATION TASKS.

Subject's Intention	*Correctly Classified*		*Incorrectly Classified*	
Sketch	45	(84%)	9	(16%)
Analysis	25	(81%)	6	(19%)
Manipulation	9	(90%)	1	(10%)

- **Finding 1: Support for refined or follow-up queries is needed**

For example, "*what are the other attributes around depth of 7.5*", "*what is the heaviest American car*". These queries are usually follow-ups from sketch questions. In the first example, a certain value or a value range is specified, like a filter in the database operation. In the second example, the query contains superlative adjective "*heaviest"* referring to the attribute weight. To link the comparative or superlative adjective to a data attribute properly, further knowledge about the data will be needed.

Our current conversational interface is more of a user-initiative system in which the user has freedom in what they say to the system, while the system remains relatively passive, asking only for clarification. As Zue et. al.[Zue00a] pointed out this may lead "the user to feel uncertain as to what capabilities exist, and may, as a consequence, stray quite far from the domain of competence of the system". To avoid such situations, a more active feedback from the system will be needed, for example suggesting related queries as well as associated results to help the users find their solution.

- **Finding 2: Metadata is needed for providing context to parsing**

Take "*compare the price of meat*" for example. "Meat" actually refers to "beef" and "chicken" in the Average U.S. Food Prices dataset. Correct recognition of such terms requires knowledge about synonyms or hypernyms of the attribute names. One solution might be using WordNet[16] to find Synset for each attribute. This also suggests in the creation of a database or table, each attribute accompanied with a meta-field briefly explaining the terms should be preferable.

Another field needed in the metadata will be the Units of the data. Unknown data units can confuse the query classification. For example, "*compare the price of apples with tomatoes*" versus "*compare cylinders with horsepower*". In the first example, apples and tomatoes have the same data unit (price), so put their values both on the y axis and use years as the x axis is more

desirable than plotting them on x, y axes respectively. But in the second example, the two attributes cylinders and horsepower have unmatched units, user would expect them plotted as x and y axes separately.

In the comparison study with Microsoft Excel, I asked the subjects for their opinions on the resulting Excel charts. The findings were highly encouraging. One participant preferred Excel's results over *Articulate*, six of them felt the results were acceptable though not ideal, and four of them found the charts completely inappropriate. Most participants found the steps needed to create a chart in Excel to be complex, and the means for selecting data ranges and series, confusing. Furthermore, I found that at least half of the subjects that used Excel had to create more than one chart and call additional Excel functions (such as sort, min, max) to describe a query that otherwise could have been expressed with a single sentence in *Articulate*. Lastly, subjects on average took twelve times longer to realize a graph for a query in Excel than in *Articulate* - which typically took less than a minute.

These initial results were promising, and shed light on a couple of issues that I was keenly interested in investigating, such as suggesting related queries, improving feature recognition by syntactic analysis. Accordingly, improvements are made in the design of *Articulate* as described in Chapter 3. Integrating these capabilities into *Articulate* can help to better understand user's intentions and generate more appropriate visualizations.

5.2. Case Study

The main contribution of *Articulate* is the proposed methodology for translating imprecise natural language queries into meaningful visualizations. To be specific, the idea is: an imprecise query is converted to a feature vector, and classified as certain visualization task, then translated to SimVL commands and presented automatically as a graph. While the initial prototype focuses on producing information visualizations, this methodology can be applied to

domain science visualizations by adjusting certain steps in the framework. The design experience for the Endurance application that follows illustrates this idea.

5.2.1. Motivation

ENDURANCE (Environmentally Non-Disturbing Under-ice Robotic ANtarctic Explorer) is a NASA funded project involving the development of an autonomous underwater vehicle (AUV) capable of generating 3-D bio-geochemical datasets in the extreme environment of a perennially ice-covered Antarctic dry valley lake. The reasons for choosing this domain are manifold. Firstly, our lab has already established a trust relationship from prior collaborations with them. They have been using the visualization tools developed in our lab to support researchers in the analysis of the data coming from the ENDURANCE mission. And the familiarity with their discipline enables our future observations to be more nuanced. Secondly, the ENDURANCE application offers a blend of statistical as well as scientific visualization problems. Lastly, based on our collective experience in collaborating with other domain science teams, this community typifies the way many scientists work.

5.2.2. System Adjustment

The first step in the development of the ENDURANCE application is to understand their scenario. Preliminary observations of interactions between an EVL visualization expert and several of the participating domain scientists using the existing visualization tool were conducted. Through the observation, several distinctive behavioral features of the domain scientists were found during their data analysis:

- Visualize data in both location-based color-coded 3D representation as well as 2D plots.

- Generate plots showing the relations between different chemical and measurement parameters, and compare multiple plots side by side. Here are a couple of excerpts:
  - *"Can you show density versus depth?"*
  - *"Temperature, salinity, conductivity. Can we do all three of them?"*
- Select or filter data by time and location. For example:
  - *"Can we chop the depth to 20 to 25 meters?"*
- Manipulate the visualization in certain ways. For instance,
  - *"Switch density to conductivity."*
  - *"Can you rotate the bathymetry?"*

Based on the above observations, I adjusted the *Articulate* system in certain aspects. The first adjustment is in the meta-data deriving step: all the biological and geochemical measurement parameters are extracted from the original data files, as well as some domain specific terms used by the scientists, such as bathymetry, slope, and scale. Secondly, feature representations are tailored to reflect the domain scenario:

- A couple of shallow linguistic cues are identified to distinguish 3D visualization request from 2D plot request: query contains particular keywords including 3D, volume, volumetric, bathymetry, bathymetric; query contains location attributes x, y and depth together.
- Constraints specified on time, such as "between December 15th and December 18th" or "after December 10th" are recognized in addition to filters on numerical data.
- Some customized manipulation commands are added to this application, including flipping axes, adding or removing an attribute from an axis, replacing an attribute in the plot.

Finally, in the Visualization Executer, adaptions are made in the graph reasoning algorithm to accommodate 3D views, mainly for the relationship task with regard to a measurement and the location. The measurement is often color-coded over the 3D glyphs with a legend describing the color scale. Also noticed in the scientists' interaction with the existing visualization tool, for the relationship task on multiple variables, a side-by-side multi-panel plot is preferred than a single plot with overlapped multiple axes (as illustrated in Figure 19). To implement 3D visualization, VTK was integrated into *Articulate's* graph engine. In the current application, only scalar data is visualized using 3D glyphing techniques. But in the future, I fully intend to extend this to accommodate complex visual representations such as isosurface or streamline visualizations. Figure 20 gives an example of the current 3D visualization for the ENDURANCE application; as shown in this example a 2D lake map with grids is overlaid on top of the 3D glyphs to highlight the location information.

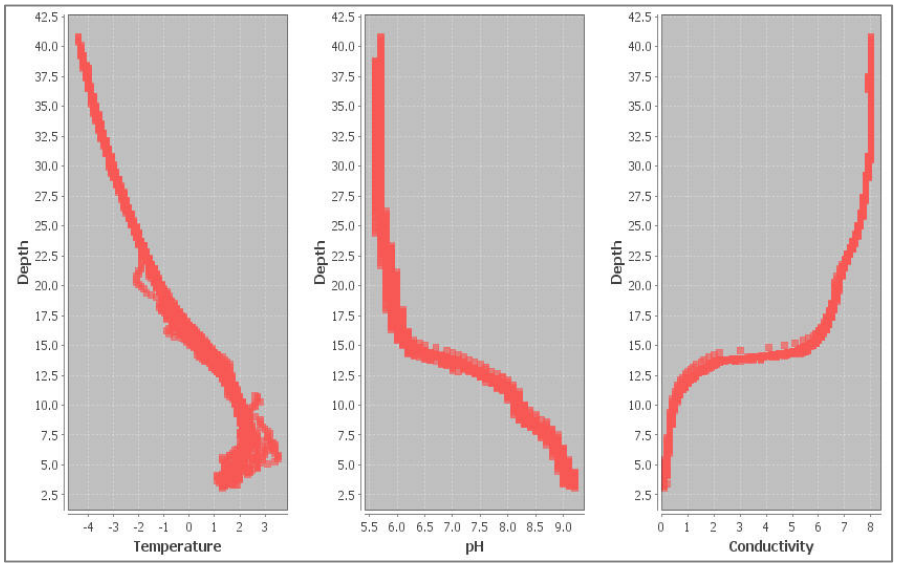

Figure 19. Visual results translated from "*can you plot depth versus temperature, pH and conductivity*".

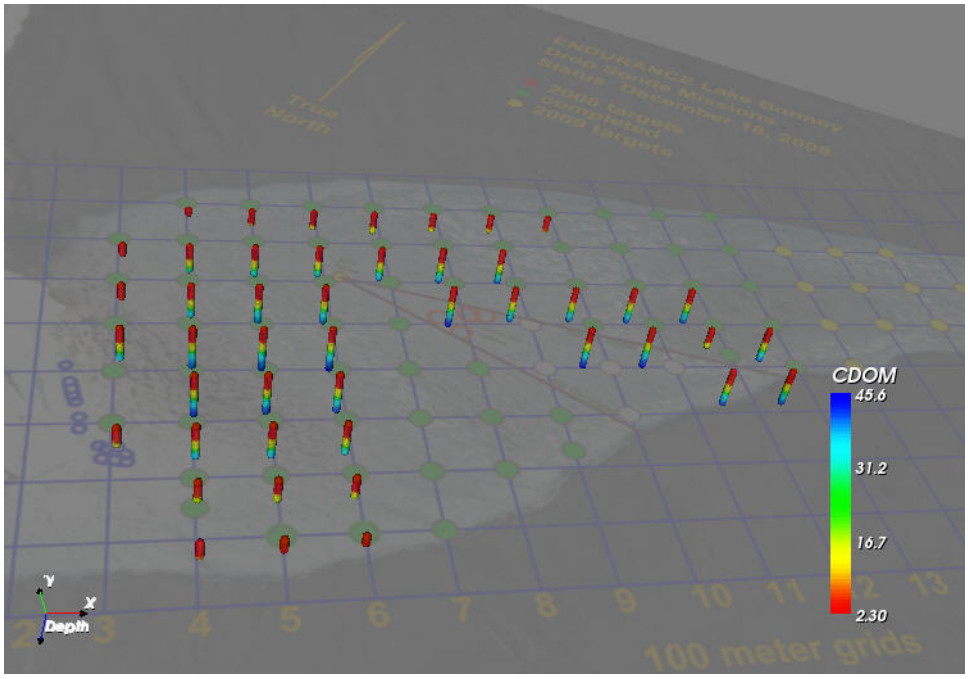

Figure 20. 3D visualization for the query "*show me the relationship between depth and CDOM in a 3D view*".

Before having access to *Articulate*, the domain scientists have to interact with a visualization expert to create various visualizations needed in their data analysis. With the help of *Articulate*, scientists will be able to use language to create visualizations and modify visualizations that have been previously created. Since the framework of *Articulate* is composed of distinct modules, each of them handles a single process such as data processing, language parsing, or graph generation; it is very flexible to be applied to different domain science by adjusting certain modules in the framework.

5.3. Formal User Study

Through the preliminary study and the ENDURANCE case study, I discovered several aspects that were in need of improvement, including suggestions for related queries and visualizations, linguistic analysis to enhance feature recognition, more support of manipulation

and data filter tasks. The current design is the result of an iterative design process that has addressed these issues. To evaluate the incremental version of *Articulate*, a formal between-subject experiment was conducted.

5.3.1. Study Methodology

This study is designed as a one-way between-subject experiment, in which I am interested in discovering whether the users using *Articulate* can: produce more appropriate visualizations and fewer irrelevant visualizations, and make discoveries that would not have occurred otherwise.

Participants

Seventeen subjects (three females and fourteen males) participated in the study. Sixteen of them were graduate students, and one was a post-doctoral researcher. The reasons for recruiting these subjects are two folds: all of them have some experience with visualization tools, such as VTK, Paraview, Gnuplot and Microsoft Excel, or have built data visualization tools. On the other hand, none of them have used Gapminder (the tool where the data set and the tasks were originally presented) before. Only one participant has seen Gapminders's website briefly and all the others have no experience with Gapminder at all. This was desirable because it avoids any preconceived notions about the relevant visualizations which may bias the evaluation of the test system.

Tasks

The task was a visual analytic problem about multi-dimensional time-series abstract data. The dataset consists of four countries (Brazil, Ghana, Japan, and Philippines) over 50 years (1950 – 2005) wealth and health development data (including 6 attributes). The data was extracted from

Gapminder World's data repository [Gapminder]. The participants were expected to perform three tasks at two different levels of complexity (two simple direct tasks and one complex open-ended task). They were asked to finish each task by exploring the data with *Articulate* and verbally answering the given question based on the visual results generated by *Articulate*. The specific tasks are shown below:

- Task 1: What are the top 2 countries that emit most CO2 recently?
- Task 2: Is there any big difference between the trends of Japan and Philippines?
- Task 3: Based on this data set which attribute(s) do you think are important factors to life expectancy?

These tasks are derived from the global development quiz provided by Gapminder World for secondary school students in their history or social studies. There are several benefits to use this scenario. First, the answers are not trivial, but the problem was designed such that participants could be expected to make reasonable progress within a reasonable amount of time (30 minutes at the most). Second, the data is not derived from a specialized domain, so no particular background knowledge is required to understand and perform the tasks. Third, questions with answers that may influence the following ones were introduced to simulate successive incremental behavior. And lastly, these tasks can be used to stimulate the interest in using statistics to understand the world. The sometimes surprising answers contained in the tasks can challenge the subject's world view.

Procedure

The study was performed in our lab using a PC running 64-bit Windows 7 system. The participants were seated in front of a 46-inch 1920 by 1080 resolution HDTV monitor and provided with a head mounted microphone for voice recording and a regular mouse for complementary interaction. The large screen real estate could encourage individuals to pose more queries, hence to produce more visual representations than what is typically possible with a

laptop display. This would give the user the advantage of having quick access to alternative perspectives of the data, thus increasing the quality of data analysis and decreasing the time required to get to a solution.

In the study, the participants were firstly given a five-minute demonstrative introduction to the *Articulate* system. It also gave them a clue of what queries could be answered by the system. Otherwise, it could be difficult for people to figure out how to formulate the first query for the tasks. Then the participants were asked to make verbal queries to the system to explore the data and summarize their findings based on the visual results showing on the screen. During the task, they were encouraged to think aloud about what kind of jobs they were doing when expressing natural language queries to the system, such as to get an initial idea of the dataset, or to filter out certain values. This necessitated the use of natural language as the major interaction modal. I am aware that even though speech recognition has made significant advances, it is still not perfect. Therefore, in this study I used a Wizard-of-Oz approach in which the investigator took the role of a speech recognition engine and transcribed spoken language into *Articulate*. The test version of the *Articulate* system has the capability of suggesting related visualization. So if the participant found a suggested visualization is better than the system selected visualization in answering the question or helping derive the answer, he or she can highlight that suggested visualization by double clicking it on the suggestion panel. Some of the questions were designed to be open-ended. The participants were not required to find the correct answer to each question nor had they been told whether their answers were correct, they were simply asked to find an answer they believed to be correct.

Data Collection

The study was videotaped to record user's actions and the contents on the display. While audio was recorded to capture user's natural language queries as well as the thinking-aloud comments. Objective measurements include: response time on each query; number and types of

queries and visualizations participants produced; number of suggestions used to approach desired result. User's intended visualization task for each query was also logged during the study and marked as the label for the query.

The subjective evaluation of each participant's behavior and performance were gathered by the investigators that involving observations during the task, observations after the task by reviewing the video recordings, and through a post-task survey. In the survey participants were asked a common set of questions about perceived ease of use of the system, the relevance of the produced visualization, perceived efficiency of suggestions, their positive and negative experiences with the system, and any suggestions they may have for improvement.

5.3.2. Quantitative Analysis

All participants completed the tasks, but their total completion time varied (mean = 26.8 minutes, SD = 9.3) and the number of queries used were different (mean = 8, SD = 3.4). To investigate the different behavior and the effectiveness of the system, I performed statistical analysis on three important factors in the experiment:

Query per Task

The first parameter I measured is the number of queries user initiated to complete each task. Since there were three different tasks, I wanted to find out if the number of queries was related with the type of task. A one-way analysis of variance (ANOVA) with $\alpha = 0.05$ was used for this analysis. The average number of queries initiated for each task was shown in Figure 21, where the error bar indicates the standard deviation. For task 1 and 2, on average about 1.5 queries were needed to solve the problem, which clearly showed the efficiency of the system. Task 3 required more queries (mean=4.8) to solve than task 1 and 2 due to its complexity. The

ANOVA result also indicated that the level of complexity of the task had a significant effect on the number of queries ($F_{2,48}$ =12.45, $p < 0.0001$).

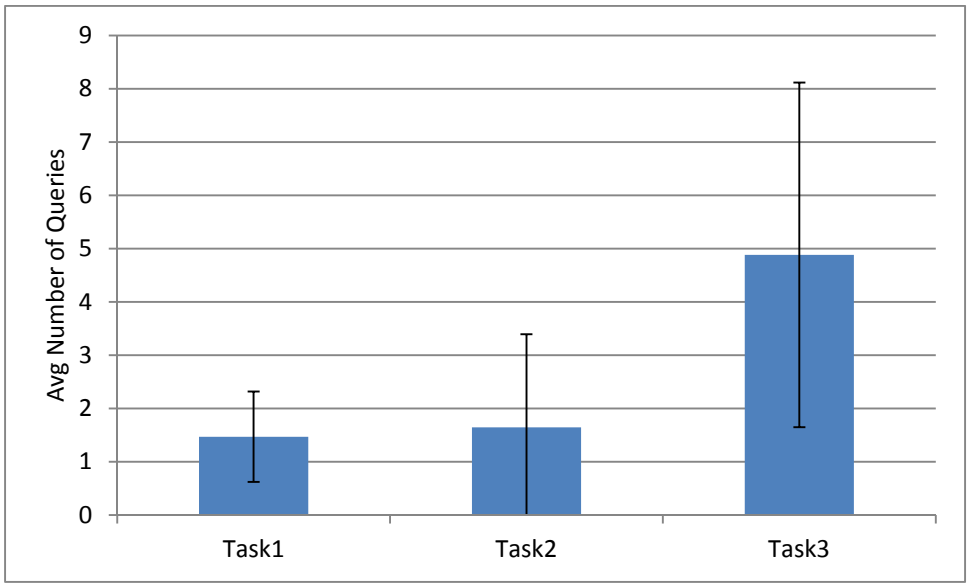

Figure 21. Number of queries for each task.

Also noticed is that the variance for each task differed dramatically. To understand this difference, I further looked at the number of queries each subject made (Figure 22). Obviously, subjects had to make more queries to complete the complex task (task 3). Because for this kind of open-ended question, the subjects could have several hypotheses in their brain; therefore, they formulated a set of queries to test them. But for task 2, it was observed that six out of seventeen subjects did not make any specific query to solve it. The reason for this was that the task 1's visual results provided deeper insight into the data which helped answer the question in task 2. And two other subjects reported completing task 2 with strong supportive information from task 1's results as well. This demonstrates the efficiency of the *Articulate* system that provides

meaningful visualizations to the user, helps them make discoveries that would not have occurred otherwise, and eventually speed up their data exploration process.

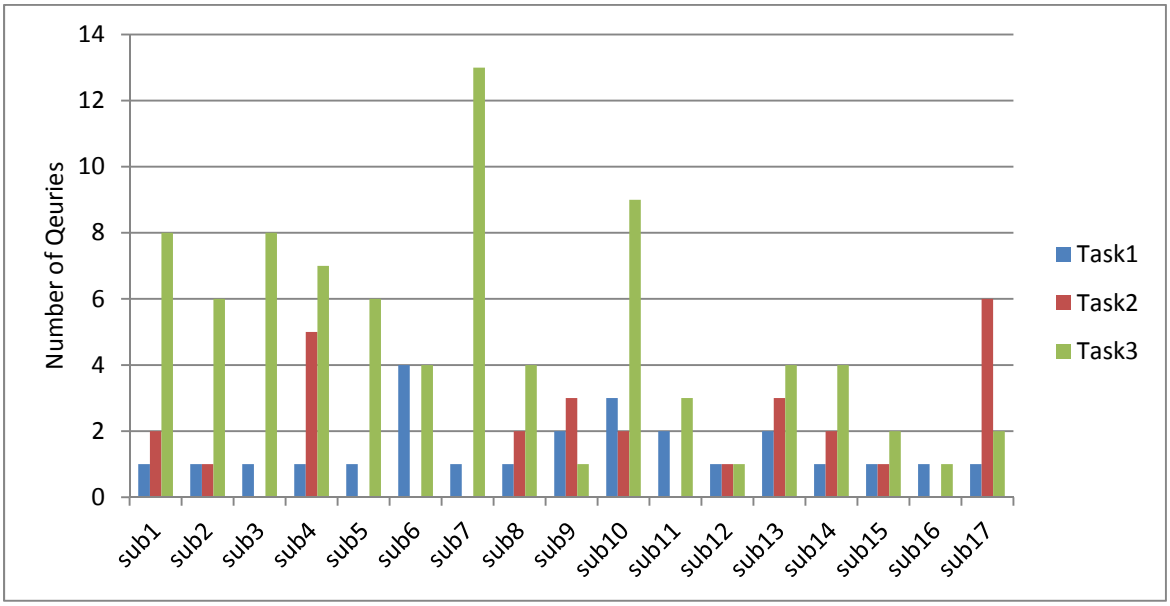

Figure 22. Number of queries each subject made per task.

Suggestion Usage

The number of suggested visualizations the subject picked over system-selected visual result was recorded. I calculated the percentage of queries in which suggestions were used, and found there was 29% chance the subjects found the suggested visualization to be more useful. This validated the efficacy of the suggestion feature in *Articulate*. I was also interested to explore if the type of tasks had any strong influence on the usage of suggestions. So ANOVA was carried out on the percentage of queries when suggestion was used for each task (Table III). The result did not show any significant difference between simple and complex tasks. Also the F-ratio ($F_{2,48}$ =0.047, p >0.05) did not exceed the critical value (F_{crit} =3.19) to reject the hypothesis that the expected values of suggestion usage in the three different tasks are similar. A likely reason for

this might be the usage of suggestion is contingent upon the purpose of the query but not the complex level of task. Hence, I performed another ANOVA with the category of user's intentions as the single factor (Table IV). This analysis revealed that suggestions were more often used in sketch tasks, where users were intended to get the overall idea of the data, than defined tasks like statistical analysis or data filtering ($F_{3,64}$ =5.27, p=0.002).

Table III. SUGGESTION USAGE AS PERCENTAGE OF QUERIES FOR EACH TASK.

Groups	*Average*	*Variance*
Task1	31%	22%
Task2	27%	16%
Task3	29%	14%

Furthermore, I analyzed the subjective rating in the post-study survey pertaining to the effectiveness of the suggestion function. The rating is based on a 5 point Likert scale (1 = Never helpful; 5 = Always helpful). The result was positive (mean=3.7, SD=0.8). The main reason subjects liked the suggestion function was that it provided alternative visualizations that gave insights on alternative perspectives of the data, which could potentially help users find their solution quickly.

Table IV. SUGGESTION USAGE AS PERCENTAGE OF QUERIES FOR DIFFERENT VISUALIZATION INTENTIONS.

Groups	*Average*	*Variance*
Sketch	21%	5%
Analysis	3%	1%
Manipulation	0	0
Filter	6%	6%

Classification Error

Since each query was labeled with the subject's intended category of visualization task, I was able to perform an analysis of the classification error, i.e. how many queries were classified in a category different from the subject's label. The results are listed in Table V. For each category of intended task there was an average of 5% classification error (SD=0.06) and it showed a significant difference among different categories of tasks. To be specific, it occurred more often in sketch tasks than other categories.

Table V. CLASSIFICATION RESULTS FOR EACH CATEGORY OF VISUALIZATION TASKS.

Subjects' intention	*Total number of query*	*Number of incorrectly classified*	*Error rate*
Sketch	108	16	14.8%
Analysis	4	0	0
Manipulation	18	1	5.6%
Filter	5	0	0

To discover the cause of this, I looked through those incorrectly classified queries, and made an interesting finding. Many of those queries were similar to the following ones:

"*Show me the CO2 emissions of Japan and Philippines.*"

"*What is the life expectancy rate between Brazil and Japan?*"

The feature vector of them showed little information: no indication of any categorical keywords; no special linguistic clues (such as using superlative or comparative adjective or adverb, quantifier or cardinal number); the type of clauses varies. The only useful information was it mentioned at least one data attribute directly (such as *CO2, life expectancy* in the above examples), and specified a data filter (*Japan and Philippines, Brazil and Japan* both are specified values for the attribute *country*). By analyzing user's feedback for such queries and the usage of suggested visualization, I found a possible approach to resolve the ambiguity – using the context. The context can be either the previous query, which often significantly influences user's preference in the subsequent query; or the domain preference, for example, in this case the data is time-series abstract value, so a comparison visualization showing the trend of data will be desirable.

5.3.3. Observations and Implications

Subjective results were assessed by examining how the participants responded to the post-study survey with questions pertaining to perceived ease of use, perceived quality of visualization, and positive and negative experiences with the system. A couple of interesting phenomena were uncovered through the examination. In this section, I first describe user acceptance to *Articulate*. Then the impact on subject's behavior and performance by using this system is discussed. Finally I conclude with users' suggestions about possible improvement in the system design.

User Acceptance

For a new interface or tool, it is most concerned whether the subject could use it and learn from it. Thus, it was pleased to find the subjects were all able to use *Articulate* successfully in this study to complete three different tasks. And the responses to the following statements in the post-test survey were positive based on a five-point Likert scale (as shown in Figure 23):

Q1: "Learning to operate the system is" Rating 1=very difficult, 5=very easy;

Q2: "Task can be performed in a straight-forward manner." Rating 1=never, 5=always;

Q3: "Visualization as you expected." Rating 1=never, 5=always;

Q4: "Visualization relates to the task." Rating 1 = always misleading, 5= mostly beneficial.

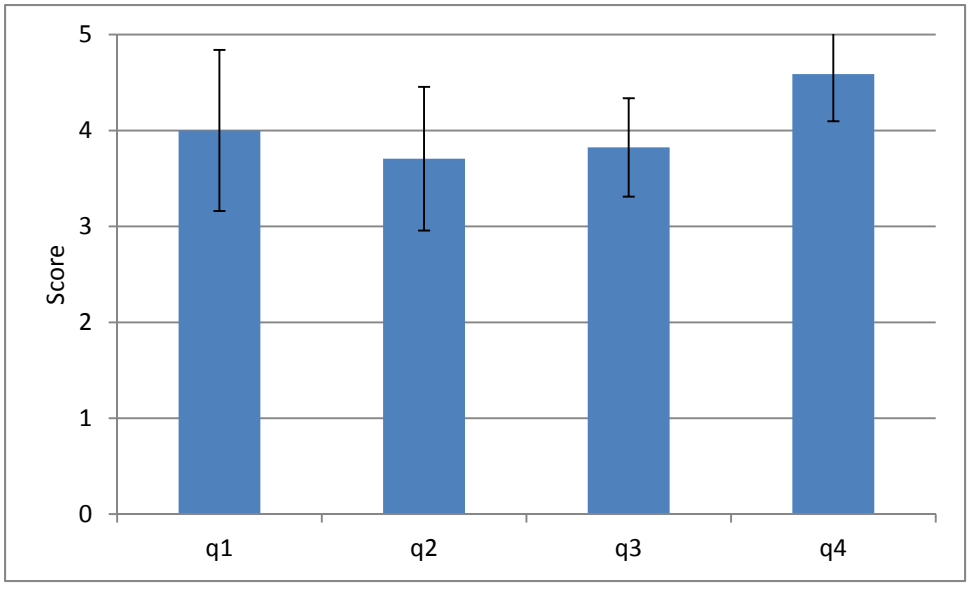

Figure 23. Average Likert scale rating on four questions in the post-test survey.

The assessment on the overall system showed that users accepted the system rapidly. And most users reported being comfortable with the technology after the study:

"This was very useful for precise graphing. Like I was able to answer (task) 1 and 2 without too much work on my part." (Subject 2)

"It was very easy to use. Understood what I was asking." (Subject 13)

"I think the best advantage here is it can make generating things faster, you don't have to go and plot it. The program more or less has an understanding of the data, you just tell it to show me this and that, and it knows how to show it." (Subject 6)

Impact on User's Behavior and Performance

Learning any new system can increase cognitive burden, and in this case, that means the transition from mind-hand coordination to mind-mouth coordination. In this study, all of the subjects were able to alter their working styles to adapt to the new environment. Their feedbacks on using *Articulate* compared with other traditional visual analytic tools showed that the subjects were more favorable towards natural language guided interface.

"I think being able to just speak my question instead of having to type it is very helpful." (Subject 17)

"It is going to be faster than Excel on (task) 3, as (in Excel) I will lose a lot of time graphing, re-graphing and trying to process the data by myself. " (Subject 2)

"In other tools like Excel I have to put in the data, find the right chart, make everything organized. But for this, I asked a question, it pulls the data for me. It gives me easier access to different types of graphs without me having to go back and fill in the place again. I found that's good." (Subject 13)

And the system had unknowingly affected user's preference over different visualizations. As one subject pointed out:

"In another case where I asked can you show me everything, and it gave me a lot of little charts. I expected all to be in one chart, kind of confusing. But it spread things out. And the separated little charts made it very easy to read."(Subject 17)

Similar to what I found in the quantitative analysis, the capability of offering suggested visualization has been enjoyed by most of the subjects, and showed potential impact on their performance.

"I like the suggestions because they were not necessarily things I was looking for myself. … Most of time it understood what I was looking for, but it also brought in things that I wasn't looking for that ended up being helpful. It did a good job translating my requests but also giving me alternatives." (Subject 7)

"I really like the suggestion idea. It gives me some other information about what is going on." (Subject 15)

"And I like that it gives you suggestions just in case it might have interpreted you wrong, and you can still choose other things." (Subject 17)

Recommended Improvements by Users

Even though some users in the beginning were concerned about the effectiveness of *Articulate* in this task, at the end, the assessment and feedback from all the subjects were very positive. They had obtained a better idea of how to perform visual analysis using natural language queries. However they also expressed some suggestions on further improvement, which were summarized into several categories:

First, starting with some visualization examples about the current data on the screen will be helpful, because it is difficult to come up with the first query.

Second, a label attached with each graph could potentially speed up the data exploratory process. On one hand, textual notes on previous visualization remind users of what they asked

previously, on the other hand, language hints on suggested graphs could help users formulate alternative queries rapidly.

Third, a strategy to display all the previous visual results more efficiently is needed. One possible approach is to line up all the existing views in thumbnails at the side of the main panel, which can reflect what the user has done in a linear fashion. Another method would be to support the tiling feature, which can provide a quick way to lay out all the windows without overlapping.

Aside from the above suggestions, users also indicated that it would have been useful if *Articulate* could also explain the graph it had produced. These are issues that I am keenly interested in investigating in the future.

6. CONCLUSION AND FUTURE WORK

This work has demonstrated an approach for the creation and modification of meaningful visualizations from user's imprecise verbal descriptions. Unlike traditional visualization tool, this approach brings together natural language processing, machine learning and visualization techniques, to enable the interaction with speech, hence relieve the users of the burden of having to learn how to use a complex interface for their data exploration.

In this dissertation, I developed *Articulate*, a generalized framework that implements the proposed approach to translate imprecise verbal descriptions into visualizations using a natural language interface. The major components of the framework include: Data Parser, Input Translator, and Visualization Executer. A Data Parser collects information from the data file and prepares them in metadata formats to provide a brief context for interpreting user's interests on the data. The Input Translator recognizes the intended data and translates user's natural language descriptions into specific visualization tasks expressed in a formal grammar. These explicit expressions together with the impact of various graphical primitives guide the Visualization Executer in determining the most appropriate graph.

Through the incorporation of a speech interface and natural language parser, users are allowed to "tell" the computer what they want to see, and have the system intelligently create the visualization. This is compelling because it can provide a rapid means of creating data plots and other more advanced visualizations without requiring the user to struggle with yet another user-interaction device. It is important to understand that the goal of this design is not to trivially translate simple queries such as: "make a plot of temperature versus pressure"- though it is certainly possible within the context of what I propose. Instead the expectation is that this approach will enable a user to ask deeper questions about data, without having to follow or memorize a strict grammar or command structure.

In the input translating process, I investigated the linguistic features of the user's query, and devised a multi-dimensional feature space to represent both syntactic and semantic characteristics of a query. The query is then converted to a feature vector, which is mapped to a visualization task by applying a supervised learning algorithm, and finally translated into explicit graphical commands. Meanwhile alternative queries and visualization are generated within the context to help users find the truly intended visualization quickly.

Formal studies via between-subject experiments were conducted to evaluate the proposed approach. *Articulate* users took less time and created fewer charts to produce a desired result as compared to using a popular graphing program. The studies also showed this approach was able to provide meaningful visualizations to the user, help them make discoveries that would not have occurred to them otherwise, and eventually speed up their data exploration process. Along with that, a case study was presented in a domain science visualization application, which illustrated how the approach was conceptually extensible to scientific visualizations as well.

This dissertation sought to understand the user's intention and interactive behaviors in the data visualization and analysis tasks, and investigate how to best take advantage of a natural language interface to facilitate these tasks. I believe the proposed methodology has the potential to help domain scientists as well as broader audience who are consumers of information, such as citizen scientists, public policy decision makers, and students, to rapidly produce effective visualizations without extensive knowledge of visualization techniques and/or tools.

The work presented in this dissertation offers a novel approach for creating visualizations from verbal descriptions automatically; however, there are still limitations need to be addressed. For example, in the current graph engine, only standard 2D graphs and plots, as well as basic 3D scalar visualizations are supported, which in the future I fully intend to extend to accommodate more advanced visualization techniques such as those commonly used in scientific visualization (for instance volumetric or streamline visualizations).

Another direction in this research is to provide a textual or verbal explanation to the user on how and why the resulting visualization was derived by the system. *Articulate* will need to generate appropriate responses whenever it recognizes that the user is asking for an explanation, which may include requests for information about how to visualize the data and requests for clarification on what the visualization represents. When answering such questions, *Articulate* needs access to its own knowledge about the implementation of the current visualization, such as the mapping from data attribute to visual primitives. Providing explanations of the visualization can help users learn visualization techniques they did not know before, and make better understanding about the data.

In terms of interaction modality, enabling gesture input as a complementary interface to voice input is a promising area to investigate. Because during natural language interaction, verbal communication is often enhanced or further clarified through a variety of different gestures. As observed in the user study, subjects tend to point and gesture to the screen when saying “this”, “these”, “here” in deictic expressions. Recognizing pointing gestures can help the interpretation of references used in the verbal description. With devices such as a multi-touch screen and optical trackers, it will be relatively easy to extend the *Articulate* framework with the gesture interaction.

The presented research results, although far from constituting a complete set, offer direction for the future investigation of the automatic generation of visualization in a natural interaction environment. As the technology needed for building such environments becomes more affordable, we will likely see natural language interfaces permeate a broad range of everyday use cases. Since the Internet has made it possible for everyday citizens to access vast amounts of data instantaneously and at very low cost. There is tremendous interest amongst researchers in the visualization community finding better ways to harness data to make it useful to individuals, businesses, and companies. The work proposed here will contribute greatly to that effort by

making it possible for non-visualization-experts to be able to leverage modern advances in visualization and help them interpret data that is growing exponentially in scale and complexity.

CITED LITERATURE

[Abela81] Abela, A. “Advanced Presentations by Design: Creating Communication that Drives Action”. Pfeiffer, 1981.

[Amar05] Amar, R., Eagan, J., and Stasko, J.. “Low-level components of analytic activity in information visualization”. In Proceedings of the IEEE Symposium on Information Visualization, 2005.

[Bavoil05] Bavoil, L., Callahan, S., Crossno, P., Freire, J., Scheidegger, C., Silva, C., and Vo, H. “Vistrails: Enabling interactive multiple-view visualizations”. In IEEE Visualization, 135–142, 2005.

[Bostock09] Bostock, M. and Heer, J. “Protovis: A Graphical Toolkit for Visualization”, IEEE Transactions on Visualization and Computer Graphics, 15(6): 1121-1128. 2009.

[Cleveland84] Cleveland, W. S., and McGill, R., “Graphical Perception: Theory, Experimentation, and Application to the Development of Graphical Methods”. Journal of the American Statistical Association. 79(387): 531-554, 1984.

[CMUSphinx] http://cmusphinx.sourceforge.net/

[Cortes95] Cortes, C. and Vapnik, V., "Support-Vector Networks", Machine Learning, 20, 1995.

[Cox01] Cox, K., Grinter, R. E., Hibino, S. L., Jagadeesan, L. J. , Mantilla, D. “A Multi-Modal Natural Language Interface to an Information Visualization Environment”. Journal of Speech Technology. 4, 297--314, 2001.

[Cunningham02] Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.. “GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications”. Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02). Philadelphia, 2002.

[Dennett92] Dennett, D., “Consciousness Explained”. Little, Brown and Co., 1991.

[Duke09] Duke, D. J., Borgo, R., Wallace, M., and Runciman, C.. “Huge Data But Small Programs: Visualization Design via Multiple Embedded DSLs”. In Proceedings of the 11th International Symposium on Practical Aspects of Declarative Languages (PADL '09). 31-45. 2009.

[Ebert08] Ebert, D., Gaither, K., Gilpin, C., “Enabling Science Discoveries Through Visual Exploration”. National Science Foundation Workshop report, Washington, D.C., 2008.

[Feiner85] Feiner, S.. “APEX: An Experiment in the Automated Creation of Pictorial Explanations”, IEEE Computer Graphics and Applications 5(11), 29-37, 1985.

[Gapminder] http://www.gapminder.org/.

[Gilson08] Gilson, O., Silva, N., Grant, P.W., and Chen, M., “From Web Data to Visualization via Ontology Mapping”, Computer Graphics Forum, 27(3): 959-966, 2008.

[Gotz08] Gotz, D., and Zhou, M.. “Characterizing Users' Visual Analytic Activity for Insight Provenance”. IEEE Visual Analytics Science and Technology (VAST), Columbus, Ohio, 2008.

[Grammel10] Grammel, L., Tory, M., Storey, M-A., “How Information Visualization Novices Construct Visualizations,” IEEE Transactions on Visualization and Computer Graphics, 16(6): 943-952, 2010.

[Grasso98] Grasso, M. A., Ebert, D. S., and Finin, T. W. “The Integrality of Speech in Multimodal Interfaces.” ACM Trans. Comput.-Hum. Interact, 5(4):303–325., 1998.

[Hanks10] Hanks, P.. “How People Use Words to Make Meanings”. Proceedings of the 7th International Workshop on Natural Language Processing and Cognitive Science, 3(13), 2010.

[Hall09] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H., “The WEKA Data Mining Software: An Update”, SIGKDD Explorations, 11(1), 2009.

[Hallett08] Hallett, C.. “Multi-modal presentation of medical histories”. In Proceedings of the 13th International Conference on Intelligent User Interfaces, 80–89, 2008.

[Harris00] Harris, R.L., “Information Graphics. A Comprehensive Illustrated Reference”. Oxford University Press, 2000.

[Heer05] Heer, J., Card, S.K., and Landay, J.A., “Prefuse: A Toolkit for Interactive Information Visualization". In Proceedings of the SIGCHI conference on Human factors in computing systems, 421-430, Portland, Oregon, 2005.

[Heer10] Heer, J., Bostock, M., “Declarative Language Design for Interactive Visualization”. IEEE Transactions on Visualization and Computer Graphics, 16(6): 1149-1156, 2010.

[Huang10] Huang, A., Tellex, S., Bachrach, A., Kollar, T., Roy, D., and Roy, N.. “Natural Language Command of an Autonomous Micro-Air Vehicle”. Proceedings of the International Conference on Intelligent Robots and Systems (IROS). Taipei, Taiwan, 2010.

[JFreeChart] http://www.jfree.org/jfreechart/.

[Kaufmann07] Kaufmann, E., and Bernstein, A.. “How Useful Are Natural Language Interface to the Semantic Web for Casual End-Users?” In Proceedings of the 6th International Semantic Web Conference, 281–294, 2007.

[Kerpedjiev01] Kerpedjiev, S., Roth, S.F., “Mapping Communicative Goals into Conceptual Tasks to Generate Graphics in Discourse”, Knowledge-Based Systems, 14(1-2): 93-102, 2001.

[Kersey09] Kersey, C., Di Eugenio, B., Jordan, P. W., and Katz, S., “Knowledge Co-Construction and Initiative in Peer Learning Interactions”. In the 14th International Conference on Artificial Intelligence in Education. Brighton, Great Britain, 2009.

[Klein03] Klein, D., and Manning, C.D., “Accurate Unlexicalized Parsing”. Proceedings of the 41st Meeting of the Association for Computational Linguistics, pp. 423-430, 2003.

[Mackinlay86] Mackinlay, J.D., “Automating the Design of Graphical Presentations of Relational Information”. ACM Trans. Graph. 5(2): 110-141, 1986.

[Mackinlay07] Mackinlay, J.D., Hanrahan, P., Stolte, C. “Show Me: Automatic Presentation for Visual Analysis”. IEEE Trans. on Visualization and Computer Graphics. 13(6): 1137-1144, 2007.

[Marneffe06] Marneffe, M.-C.D. , MacCartney, B., Manning, C.D., “ Generating Typed Dependency Parses from Phrase Structure Parses”. In Proceedings of LREC-06, pp. 449–454, 2006.

[Miller95] Miller, G.A., “WordNet: A Lexical Database for English”, Communications of the ACM, 38(11): 39-41, 1995.

[Patel09] Patel, N., Agarwal, S., Rajput, N., Nanavati, A., Dave, P., and Parikh, T.S., “A Comparative Study of Speech and Dialed Input Voice Interfaces in Rural India”. In Proceedings of the 27th international conference on Human factors in computing systems ACM, 51-54, 2009.

[Pearl85] Pearl, J., "Bayesian Networks: A Model of Self-Activated Memory for Evidential Reasoning". In Proceedings of the 7th Conference of the Cognitive Science Society, University of California, Irvine, CA, pp. 329-334. 1985.

[Petrov06] Petrov, S., Barrett, L., Thibaux, R., and Klein, D.. “Learning Accurate, Compact, and Interpretable Tree Annotation”. In Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics (ACL-44), pp. 433-440, 2006.

[Quinlan86] Quinlan, J. R.. “Induction of Decision Trees”. Machine Learning. 1(1):81-106, 1986.

[Quinlan93] Quinlan, J. R., “C4.5: Programs for Machine Learning”. Morgan Kaufmann, 1993.

[Rezk06] Rezk-Salama, C., Keller, M., Kohlmann, P., “High-level User Interfaces for Transfer Function Design with Semantics”. IEEE Trans. on Visualization and Computer Graphics 12, 1021–1028, 2006.

[Roth94] Roth, S.F., Kolojejchick, J., Mattis, J., and Goldstein, J., “Interactive Graphic Design Using Automatic Presentation Knowledge”. In Proceedings of the SIGCHI conference on Human factors in computing systems (ACM CHI), 112-117,1994.

[Santos09] Santos, E., Lins, L., Ahrens, J., Freire, J., Silva, C., “VisMashup: Streamlining the Creation of Custom Visualization Applications”, IEEE Transactions on Visualization and Computer Graphics, 15(6): 1539-1546, 2009.

[Schulman09] Schulman, D., and Bickmore, T. “Persuading users through counseling dialogue with a conversational agent”. Proceedings of the 4th International Conference on Persuasive Technology, 1-8, 2009.

[Seneff00] Seneff, S., Polifroni, J., “Dialogue Management in the Mercury Flight Reservation System”. In ANLP/NAACL 2000 Workshop on Conversational Systems, 11-16, 2000.

[Sherwani07] Sherwani, J., Ali, N., Tongia, R., Rosenfeld, R., Memon, Y., Karim, M., Pappas, G., "HealthLine: Towards Speech-based Access to Health Information by Semi-literate Users". In Proc. Speech in Mobile and Pervasive Environments, Singapore, 2007.

[Shneiderman96] Shneiderman, B., "The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations". In Proceedings of the IEEE Symposium on Visual Languages, 336-343, Washington. IEEE Computer Society Press, 1996.

[Shneiderman00] Shneiderman, B., "The Limits of Speech Recognition". Communications of the ACM, 43(9):63-65, 2000.

[SRN09] Silicon Republic News, http://www.siliconrepublic.com/news/article/14758/randd/googles-top-inventor-says-talkingcomputers-are-the-future.

[Stolte02] Stolte, C., Tang, D., Hanrahan, P.. "Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases". IEEE Trans. on Visualization and Computer Graphics, 8(1):52-65. 2002.

[Sun10] Sun, Y., Leigh, J., Johnson, A., and Lee, S., "Articulate: A Semi-automated Model for Translating Natural Language Queries into Meaningful Visualizations", Proceedings of 10th International Symposium on Smart Graphic, pp. 184-195, 2010.

[Tufte83] Tufte, E.R. "The Visual Display of Quantitative Information". Graphics Press, Cheshire, Connecticut, USA, 1983.

[VTK] http://www.vtk.org/.

[Watson11] http://www-03.ibm.com/innovation/us/watson/index.html

[Wehrend90] Wehrend, S., and Lewis, C., "A Problem-Oriented Classification of Visualization Techniques", Proceedings of the 1st conference on Visualization, IEEE, 139-143, 1990.

[Wickham09] Wickham, H.. "Ggplot2: Elegant Graphics for Data Analysis", Springer. 2009.

[Wilkinson05] Wilkinson, L.. "The Grammar of Graphics", 2nd edition, Springer. 2005.

[Witten11] Witten, I. H., Frank, E., Hall, M. A., "Data Mining: Practical Machine Learning Tools and Techniques", 3rd ed., Morgan Kaufmann, 2011.

[Wolfram] Wolfram Research, http://www.wolframalpha.com.

[Zhou98] Zhou, M., and Feiner, S.. "IMPROVISE: Automated Generation of Animated Graphics for Coordinated Multimedia Presentations". Second International Conference on Cooperative Multimodal Communication (CMC 1998),43-63, 1998.

[Zue00a] Zue, V., Glass, J.R. "Conversational Interfaces: Advances and Challenges". In Proceedings of the IEEE, 1166-1180, 2000.

[Zue00b] Zue, V., Seneff, S., Glass, J.R., Polifroni, J., Pao, C., Hazen, T.J., Hetherington, L. "JUPlTER: A Telephone-Based Conversational Interface for Weather Information". IEEE Transactions on Speech and Audio Processing. 8: 85-96, 2000.

VITA

NAME Yiwen Sun

EDUCATION

2005 - present PhD, Computer Science, University of Illinois at Chicago (UIC)

1999 - 2003 Bachelor of Engineering, Computer Science and Technology, Shandong University, Jinan, China

PROFESSIONAL EXPERIENCE

2007 - present	Research Assistant	Electronic Visualization Lab, UIC
Summer 2011	Research Intern	Microsoft Research
Summer 2008	Research Intern	IBM T.J.Watson Research Center
2005 - 2007	Teaching Assistant	Department of Computer Science, UIC
2003 - 2005	Research Assistant	VR & HCI Lab, Shandong University

PUBLICATIONS

- Leigh, J., Johnson, A., Renambot, L., Peterka, T., Jeong, B., Sandin, D., Talandis, J., Jagodic, R., Nam, S., Hur, H., Sun, Y. "Scalable Resolution Displays". Proceedings of the IEEE, Special Issue: Beyond HDTV, 2012.
- Sun, Y., Fisher, D., and Drucker, S., "Data Narratives: Telling Stories about Data". Workshop on Telling Stories with Data: The Next Chapter, IEEE VisWeek, 2011.
- Sun, Y., Leigh, J., Johnson, A., and Lee, S., "*Articulate*: a Semi-automated Model for Translating Natural Language Queries into Meaningful Visualizations". Proceedings of 10th International Symposium of Smart Graphics, Lecture Notes in Computer Science, 2010, Banff, Canada.
- Sun, Y., Leigh, J., Johnson, A., and Chau, D., "*Articulate*: a Conversational Interface for Visual Analytics". IEEE Symposium on Visual Analytics Science and Technology, 2009, Atlantic City, NJ.
- Galgani, F., Sun, Y., Lanzi, P., and Leigh, J., "Automatic analysis of eye tracking data for medical diagnosis". Proceedings of IEEE Symposium on Computational Intelligence and Data Mining, 2009,

Nashville, TN.

- Sun, Y., Meng, X., and Yang, C., "Virtual Assembly Based on Gestures". Journal of System Simulation*,* Proceedings of Chinese Conference on Virtual Reality and Visualization, 2003, Beijing, China.

HONORS AND AWARDS

2009	Graduate Student Council Travel Award, UIC
2006	Women in Science & Engineering Program Travel Grant, UIC
2003	Excellent Bachelor's Thesis, Shandong University
2002	President Scholarship, Shandong University
2001	Second Prize in the National Mathematical Contest in Modeling, China
2000-2002	Student Leadership Award, Shandong University

PROFESSIONAL MEMBERSHIP

2008 - present	IEEE Student member
2006 - present	ACM Student member