Articulate: a Semi-automated Model for Translating Natural Language Queries into Meaningful Visualizations

Yiwen Sun, Jason Leigh, Andrew Johnson, and Sangyoon Lee

Electronic Visualization Laboratory, University of Illinois at Chicago,
{ysun25,spiff,ajohnson,slee14}@uic.edu

Abstract. While many visualization tools exist that offer sophisticated functions for charting complex data, they still expect users to possess a high degree of expertise in wielding the tools to create an effective visualization. This paper presents *Articulate*, an attempt at a semi-automated visual analytic model that is guided by a conversational user interface to allow users to verbally describe and then manipulate what they want to see. We use natural language processing and machine learning methods to translate the imprecise sentences into explicit expressions, and then apply a heuristic graph generation algorithm to create a suitable visualization. The goal is to relieve the user of the burden of having to learn a complex user-interface in order to craft a visualization.

Keywords: visual analytics, natural language processing, conversational interface, automatic visualization

1 Introduction

Much has been investigated on the design of sophisticated visual analytic 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 domain experts with marginal knowledge of 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 analysis tool, such as specifying a desired graph type for a given data set, or assigning proper data fields to certain visual parameters. To facilitate the use of advanced visualization tools by domain experts, we propose a semi-automated visual analytic model: *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.

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 [1]. Furthermore, a 2007 National Science Foundation workshop report on "Enabling Science Discoveries through Visual Exploration" [2] noted that "there is a strong desire for conversational interfaces that facilitate a more natural means of interacting with science." Scientists

frankly do not have the time or patience to learn complex visualization tools. Zue [3] 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 over a hundred millennia as compared to computers which have only become widely available since the mid-80s. And while computing software and user-interfaces will become obsolete over time, the use of spoken dialog as the primary form of human communication is unlikely to become obsolete. This inspired us to adopt a conversational interface in the semi-automated visual analytic model. 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 visual analytics tools, and thereby expanding the population of users who can successfully conduct visual analyses.

In this paper, we propose a two-step process to translate a user's verbal description to a representative visualization: first, parse the imprecise queries into explicit commands using natural language processing and machine learning methods; then, determine an appropriate type of visualization based on the commands and data properties automatically. In this initial work we limit the types of visualizations that can be generated to standard 2D graphs and plots. However in the future we fully intend to extend this to accommodate complex visual representations such as those commonly used in scientific visualization (such as volumetric or streamline visualizations).

The primary contributions of this research include: 1) the incorporation of a conversational interface and natural language parser to allow for natural language input rather than grammar based commands; 2) the development of an algorithm to automatically generate graphs based on the classification of visual analytic tasks; 3) the development of a Simplified Visualization Language (SimVL) as an intermediate expression for a user's specification, which is precise, and easy to convert in a representative graph.

The remainder of the paper is organized as follows. We discuss related work in Sect. 2. In Sect. 3, we introduce the natural language parser and the proposed algorithm for graph generation. The implementation details of the system are presented in Sect. 4. Then in Sect. 5, we present a preliminary user study and its findings. Finally, we conclude in Sect. 6, where we outline directions for future work.

2 Related Work

In the last decade several approaches for automatic visualization have been proposed. One of the first was Show Me [4], an integrated set of user interface commands and defaults that automatically generate visual presentations based on the VisQL specification language. Users place data fields into columns and rows in the interface panel to specify VisQL commands. In order to generate insightful visualizations, an understanding of the relationships between columns and rows is needed. Rezk-Salama et al. demonstrated a semantic model for automatic transfer function editing in volume rendering applications [5]. 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 [6], which simplifies 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, our approach 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.

A number of speech-based interfaces have been developed that help users to access information using a conversational paradigm. JUPITER [7] for example allows users to obtain worldwide weather forecast information over the phone using spoken dialog. It is a mixed initiative system [3] 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 interface, such as travel planning [8], health information access [9]. But few of the systems we surveyed targeted the problem of visual analytics.

Cox et. al.'s work [10] 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.

Though the idea of using a conversational interface to craft visualizations is not new, our work addresses a growing trend toward in this area, as witnessed in the recent work of Wolfram Research [11].

3 Design and Methodology

The purpose of the *Articulate* system is to process a user's imprecise description and generate a meaningful visualization that best answers their question. Figure 1 gives an overview of our model.

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 parser will interpret the sentence as a request for a relationship graph with two attributes assigned as the x and y axes. This interpretation is expressed as SimVL commands and entered into the graph generator, 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 reasoner uses this information to determine that the most appropriate visualization will be a scatter plot.

The essential parts of our framework are two steps: input parser and graph reasoner. In Sect. 3.1 to 3.3 we describe how the imprecise query is interpreted, followed by the algorithm for graph generation in Sect. 3.4.

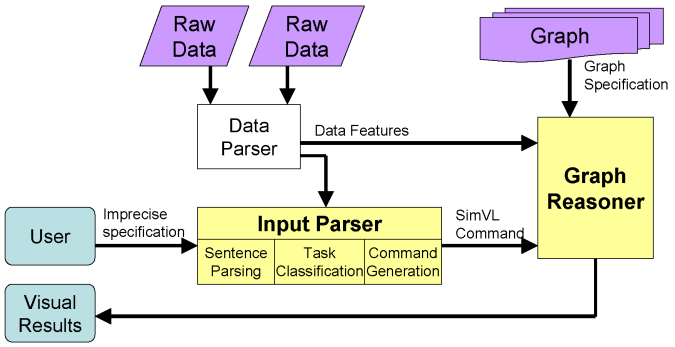

Fig. 1. Overview of the *Articulate* System

3.1 Parsing

The translation of user's imprecise specification is based on a natural language parser imbued with machine learning algorithms that are able to make reasoned decisions automatically. User's initial input to the system is a natural language query. The query sentence is parsed into a sequence of words tagged with part-of-speech labels using the Stanford Parser [12]. These labels mark 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 was conductivity related with depth*" will be tagged as:

how/WRB; was/VBD; conductivity/NN; related/VBN; with/IN; depth/NN

Using these tags, we can distinguish the functions of each word and apply different rules based on that. 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 be conductivity relate with depth

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.

3.2 Classification

Based on the parsing results, we map the query into a smaller feature space, and apply a supervised learning method in this space to predict the class of the task.

The feature space is defined as a nine-dimensional space. Each dimension describes one feature of the query, such as the existence of keywords for one class, 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_parameter_keyword [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. The keywords in each dictionary are selected according to empirical knowledge. For example, *associate, correlate, link, relate, relevant* are often used in the queries intended for 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.

The last feature can have multiple values: 0 represents no attribute names appeared in the query, 1 represents only one attribute, 2 represents two attributes, 3 represents more than two. For calculation of this feature, the original words not the stems are evaluated with existing attribute names. Because in this scenario, each attribute name is regarded as a special property noun and usually appears as a column in the data file, it has to keep its surface form, but stemming may change the form of a word, for example, "women" will be stemmed as "woman", which will bias the comparison result.

After extracting the feature vector, we use a decision tree learning method to classify the query. Inspired by Abela's chart chooser [13], we identify seven classes: comparison, relationship, composition, distribution, statistics, manipulation, and other. The first six classes are chosen based on the characteristics of different visual analytics tasks. For example, the relationship task focuses on discovering the correlation between two or more attributes; the comparison task often aims at exploring the trend over time or categories. The decision tree is built upon a set of rules obtained by applying labeled training datasets. Upon receiving the feature vector, the tree will calculate the expected values of competing alternatives. The one with the highest probability is the predicted class of the query. If "other" is the predicted class, which means the current system does not have a solution for that, or could not decide on which class the query falls in closely, then a clarification message will be presented to the user. For example if the user asked "*which car should I buy?*" the system will response with "*I'm not sure how to answer that. Can you try asking that in a different way?*"

3.3 SimVL Command Generation

To pass the classification results as well as the specified attributes to graph reasoner precisely, we propose a Simplified Visualization Language (SimVL). SimVL is specified in a formal grammar, in which each command corresponds to a specific assignment. The purpose of using SimVL is to provide a standard and simplified format for user's request in order to facilitate the graph reasoning. To accommodate the different purposes of visual analytics, this language is divided into three major categories: sketch command, analysis command and manipulation command.

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

$$\begin{aligned} <\mathit{sketch}> ::= {} & \mathit{PLOT} <\mathit{class\_of\_task}> \\ & | \ \mathit{PLOT} <\mathit{class\_of\_task}> \mathit{OVERTIME} \end{aligned}$$

As shown above, a statement is composed of an action with one or two parameters. The first parameter indicates the classified visual analytic task type, including RELATIONSHIP, COMPARISON, COMPOSITION and DISTRIBUTION. 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, mean, median and etc. so the commands are defined as an action followed by two parameters: one indicates the feature, which can be MIN, MAX, MEAN, MEDIAN, RANGE; the other lists the attribute names:

$$<\mathit{analysis}> ::= \mathit{ANALYSE} <\mathit{feature}> \mathit{OF} <\mathit{attribute\_list}>$$

Manipulation commands are used to alter some common visual metaphors, such as axis, size and color of data points.

$$\begin{aligned} <\mathit{manipulation}> ::= {} & \mathit{SETX} <\mathit{attribute\_list}> \\ & | \ \mathit{SETY} <\mathit{attribute\_list}> \\ & | \ \mathit{SET\ SIZE\_OF\_POINT\ TO} <\mathit{attribute}> \\ & | \ \mathit{SET\ COLOR\_OF\_POINT\ TO} <\mathit{attribute}> \end{aligned}$$

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.

3.4 Graph Reasoner

The final phase is the generation of the graph. Each plot type has its own advantages and disadvantages for certain types of data. For example: bar charts are often used for comparing values by category, whereas pie charts are good at illustrating relative magnitudes or frequencies. Just as a visualization expert might weigh up the pros and cons of different graphs in the determination of a desired plot type, the graph reasoner works as a smart agent carrying out a similar reasoning process autonomously. The rules

of reasoning can be divided into three categories corresponding to the three SimVL command types.

Analysis commands usually focus on the statistical features of data. So a box-and-whisker chart [14] 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 2 shows an example.

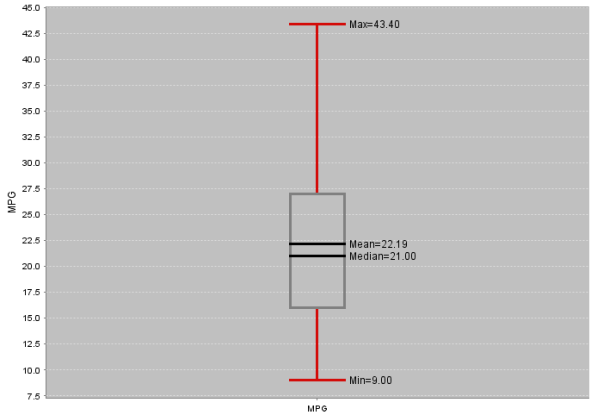

Fig. 2. Result for analysis commands translated from "*what is the range of MPG*".

Manipulation commands are typically follow-ups from a previous graph, such as switch axes, change color of point based on values of another attribute. Hence, the reasoner only needs to recognize the visual metaphor from the SimVL command, and map the specified attribute onto that. Figure 3 gives an example.

The most complicated part is the sketch commands. Figure 4 illustrates the algorithm of how the decision for the final graph is formed for these commands. There are generally four sub-algorithms for the four classes in sketch commands. Let us look at an example, given a car dataset, a query "*how has MPG changed over the years*" will be translated as SimVL commands:

PLOT COMPARISON OVERTIME

SETY MPG

The first command indicates the query is a comparison task, so we will follow the comparison sub-algorithm. It then checks whether the independent variable is ordinal. Since there is an OVERTIME parameter in the command, the independent variable will be year, thus the answer for this test is "Yes". Next step, it will look into the dataset to find whether there is a unique *MPG* value on each year. Unfortunately it does not hold. So the final graph will be a line chart with whisker (as shown in Fig. 5). Similar to the box-and-whisker plot, the top and bottom ends of whiskers indicate the maximum and minimum *MPG* values for each year.

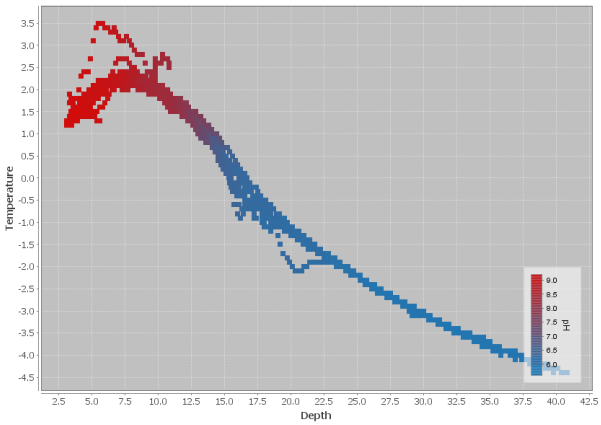

Fig. 3. Result for manipulation commands translated from "*can you color by pH*" following a query "*what is the correlation between depth and temperature*".

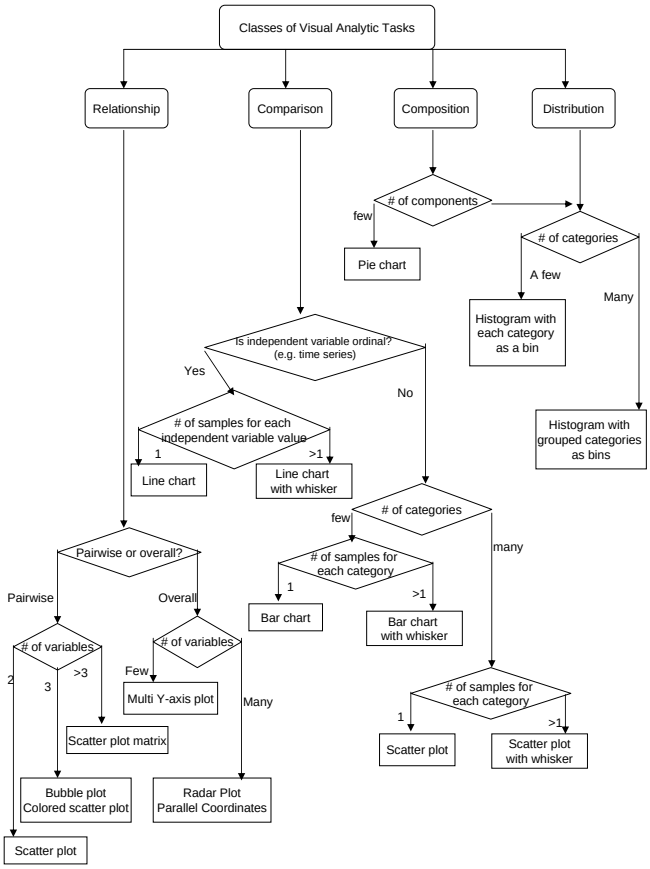

Fig. 4. Algorithm of the graph generation for sketch commands.

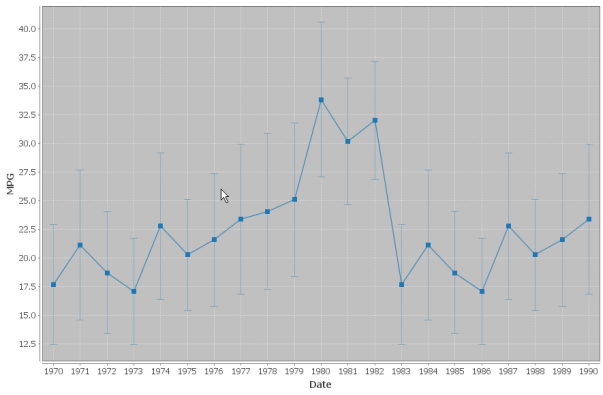

Fig. 5. Result for sketch commands translated from "*how has MPG changed over the years*".

4 System Implementation

The *Articulate* system is developed in Java. Users can input their queries by speech or text. To recognize speech we use Chant Speech Kit. Based on Microsoft SAPI, this kit translates a spoken query into a plain-text sentence. The structure of the sentence is parsed using the Stanford Parser and outputted as words with part-of-speech tags. The techniques used for query classification are based on the C4.5 system [15], which is a decision tree learning algorithm. The last component of the system is the graph engine. We adopt JFreeChart, a free Java chart library that supports a wide range of chart types. In the visualization panel, traditional mouse interaction is also supported, such as zooming and brushing, tooltips on pointing, as a complement to natural language interface for those types of tasks that are easy to perform with a GUI but cumbersome to speak. In the input panel, 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 (Fig. 6).

5 Evaluation and Results

To explore the limits of the *Articulate* system, we conducted a preliminary user study with eight graduate students. All of them are computer science majors and have some experience with visualization tools. Each of them was presented with one or two of the following datasets: a hydrologic dataset which contains 10 attributes and 4000 instances; Cars dataset found from XmdvTool website, 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.

In this study, users were asked to interact with the *Articulate* system by expressing natural language queries to finish three tasks within 20 minutes:

- Find meaningful correlations among the data attributes.
- Find as many trends as possible about the data.
- Find other interesting features about the data.

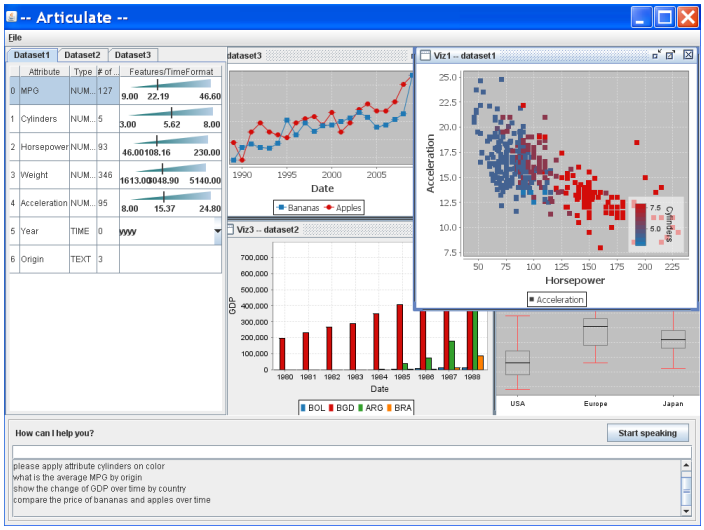

Fig. 6. Graphical user interface for the *Articulate* system. The bottom of the interface is where the user's spoken queries are displayed. The individual graphical windows depict the visualizations created as a result of the queries.

After each query, users were asked to identify their purpose of their query based on six categories: comparison, relationship, composition, distribution, analysis, manipulation. A comparison of their choice and the resulting classification is shown in Table 1:

Table 1. Results for classification. Numbers for comparison, relationship, composition and distribution categories are merged into a sketch category.

Purpose	Correctly Classified		Incorrectly Classified	
Sketch	50	(81%)	12	(19%)
Analysis	20	(87%)	3	(13%)
Manipulation	9	(90%)	1	(10%)

All subjects in our study were able to use the current system to find some data features and trends. But as shown in the table, for each category of query there is an average of 14% classification error. To discover the cause of this, we looked through each incorrectly classified query, and made a couple of interesting findings:

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.[3] 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 user 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 term 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.

Next, we repeated the study using Microsoft Excel. Eleven graduate students participated in this study. 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 had occasional experience with them, and one had no experience. Each of the subjects were given six queries, which were picked from the correctly classified queries in the first study. The subjects were asked to plot the results using Excel. After that we asked the subjects for their opinions on the resulting Excel charts. We found that one user liked Excel's result, six of them felt the results were acceptable though not ideal, and four of them found the charts completely inappropriate. Most subjects found the steps needed to create a chart in Excel to be complex, and the means for selecting data ranges and series, confusing. Furthermore, we 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 describe a query in Excel than in *Articulate*– which typically took less than a minute. These initial results were encouraging, and we intend to conduct a larger study with domain scientists in the future.

6 Conclusions and Future Work

In this paper, we presented *Articulate*, a novel visual analytic approach that utilizes a natural language interface to translate imprecise verbal descriptions into meaningful visualizations. We integrated natural language processing and automated graph generation algorithm to make the implicit sentence semantics explicit in the final representa-

tion. We believe this approach has the potential to help scientists as well as laypeople, including educators and policymakers, to quickly produce effective visualizations without extensive knowledge of visualization techniques and/or tools.

Future directions in this research will include more active reaction from the system, such as suggestion of related queries in response to unclassified questions; support for a wider range of manipulation commands, including sort, pick; and the evaluation of the system using populations that are non-computer scientists. Furthermore we hope to extend this approach to also encompass more advanced visualization techniques such as those commonly used in scientific visualization (for example, the visualization of volumetric data as isosurfaces, or vector data as streamlines).

Acknowledgments. This project was funded in part by National Science Foundation grants CNS-0703916 and CNS-0420477.

References

1. Silicon Republic News, http://www.siliconrepublic.com/news/article/14758/randd/googles-top-inventor-says-talking-computers-are-the-future
2. Ebert, D., Gaither, K., Gilpin, C.: Enabling science discoveries through visual exploration. NSF Workshop report, Washington, D.C. (2007)
3. Zue, V., Glass, J.R. : Conversational Interfaces: Advances and Challenges. In: Proceedings of the IEEE, pp. 1166–1180 (2000)
4. Mackinlay, J.D., Hanrahan, P., Stolte, C.: Show me: Automatic presentation for visual analysis. IEEE Trans. on Visualization and Computer Graphics. 13, 1137–1144 (2007)
5. Salama, C.R., 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)
6. Santos, E., Lins, L., Ahrens, J., Freire, J., Silva, C.: VisMashup: Streamlining the Creation of Custom Visualization Applications. IEEE Trans. on Visualization and Computer Graphics. 15, 1539–1546 (2009)
7. 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 Trans. on Speech and Audio Processing. 8, 85–96 (2000)
8. Seneff, S., Polifroni, J.: Dialogue management in the Mercury flight reservation system. In: ANLP/NAACL 2000 Workshop on Conversational systems, pp. 11–16 (2000)
9. 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)
10. Cox, K., Grinter, R. E., Hibino, S. L., Jagadeesan, L. J. , Mantilla, D.: A Multi-Modal Natural Language Interface to an Information Visualization Environment. J. of Speech Technology. 4, 297–314, (2001)
11. Wolfram Research, www.wolframalpha.com
12. The Stanford Parser, http://nlp.stanford.edu/software/lex-parser.shtml
13. Abela, A.: Advanced Presentations by Design: Creating Communication that Drives Action. Pfeiffer (2008)
14. Tukey, J. W.: Exploratory Data Analysis. Addison-Wesley, Reading, MA. (1977)
15. Quinlan, J. R. C4.5: Programs for Machine Learning. Morgan Kaufmann (1993)
16. WordNet, http://wordnet.princeton.edu/