

# Using Language as an Accessibility Tool: A System for Natural Language Interaction with Graphs over the Web

Leo Ferres Petro Verkhogliad Gitte Lindgaard

Human-Oriented Technology Lab  
Carleton University

Ottawa, ON, Canada

lferres@ccs.carleton.ca

## Abstract

Graphical information (such as line graphs, bar charts, etc.) are not readily available for people using mobile devices and for people who are blind or have some other visual impairments. In this paper, we report on both a general-purpose architecture to implement Natural Language Interfaces (NLIs) and an implementation of that architecture, the *iGraph* system, which allows for description and exploration of these data graphs by an NLI.

## Introduction

Graphs representing numerical information (line graphs, bar charts, etc.) are ubiquitous. In 1998, researchers calculated that at least 80% of business and other kinds of reports contain these kinds of graphs (Mittal *et al.* 1998). Unfortunately, visual interfaces are not always useful for presenting and interacting with data (e.g. for blind persons). Thus, our problem lies in delivering graphical information to people in a variety of context without relying on the visual faculties of the user.

To this effect, we have developed *iGraph*. *iGraph* is a computer system whose main objective is to allow blind and visually-impaired users gain access to graphical representations of time-series data (Ferres *et al.* 2006a; 2006b). *iGraph*'s interface is based on Instant Messaging (IM) technology: GoogleTalk, specifically. *iGraph* works by first receiving a Unified Resource Locator (URL) of an eXtensible Markup Language (XML) representation of a graph that implements our graph schema. Subsequently, *iGraph* loads the graph data in a format that allows inference and the user interacts with this information by means of a dialogue-based Natural Language Interface (NLI) with graphs as its domain of knowledge.

### *iGraph* Architecture: Messages and Managers

The *iGraph* architecture proposed in these pages is built upon two central components inspired by the literature: *messages* (Seneff *et al.* 1998) and *managers*. Managers are systems that oversee the behavior of the possibly many, and possibly redundant, systems that they implement. The XML syntax was chosen to encode the mes-

sages. Message objects are passed to all the existing subsystems for processing. Messages are created and controlled by the MessageManager (see Figure 1 for an example of the Message object representation).

In *iGraph*, we have implemented four components: (1) the LanguageManager (LgeMgr), a means to analyze the input string and generate the output string; (2) the InferenceManager (InfMgr), a means to query knowledge bases; (3) the DialogueManager (DlgMgr), a means to manage dialogue phenomena (greetings, requests, etc.); (4) the MessageManager (MsgMgr), a system that orchestrates the behavior of the three previous systems. LgeMgr, InfMgr and DlgMgr receive and send instantiations of Message to their respective subsystems. The message-passing behavior is coded in the respective Manager objects.

To exemplify the kind of processing that *iGraph* does in the background, given a graph and the NL question "what is the maximum?", the system will produce a message shown in Figure 1. The string is input into the MsgMgr, which creates an object with a MsgMgr <part> stating the provenance of the part, the ID, the user and the NL string to process. As the history shows, the message is then passed to the LgeMgr (id=1), where it passes through a PrologParser module, a CCGParser module, a CFGParser module and finally a URLParser module. Of these four parsers, only the CFGParser yields a parse, contained within the <parse> tags. Given the usually highly proprietary tags and representational media of the different parsers, each of them implement their own semantic engine, which takes a given parser's representation and returns a semantic representation in Prolog, our inference engine of choice. The semantics of the question is the query  $\max(P, 1, Z, Y)$ , which asks what the maximum value of the loaded Prolog database is for series 1 of the graph. This message then travels to the PrologInferenceEngine module, where the query to the knowledge base is actually made. In this case, the query yields one result, represented within the <inference-result> / <answerset> tag path. LgeMgr generates the NL string from a set of templates.

### A Sample *iGraph* Interaction

The interactive process works as follows: a user, connected to a given GoogleTalk account, asks the system to load a particular graph representation by means of a given URL. Once

```

<message>
  <part from="MessageManager" id="0">
    <username>xxx@gmail.com</username>
    <string>what is the maximum?</string>
  </part>
  <part from="LanguageManager" id="1">
    <parser-result from="PrologParser" />
    <parser-result from="CCGParser" />
    <parser-result from="CFGParser" />
    <parse>
      <query>
        <token name="COP1">is the</token>
        <token name="PREDWORD1">maximum</token>
      </query>
    </parse>
    <semantics>max(P,1,Z,Y)</semantics>
  </parser-result>
  <parser-result from="URLParser" />
</part>
<part from="PrologInferenceEngine" id="2">
  <inference-result>
    <query>max(P,1,Z,Y)</query>
    <answerset>
      <answer value="3" variable="P" />
      <answer value="2003" variable="Z" />
      <answer value="13.0" variable="Y" />
    </answerset>
  </inference-result>
</part>
<part from="LanguageManager" id="3">
  <generated-string>The maximum is 13.</generated-string>
</part>
</message>

```

Figure 1: Background processing of the question “what is the maximum?”, given a random graph.

the information is loaded, the user may begin to question the graph by several possible “exploration” questions, or by a “describe” command that presents a summary of the graph (Ault *et al.* 2002). All the responses from the system are read back by a text to speech engine (FreeTTS).

Of particular interest in this interaction are the properties of the generated descriptions, after the “can you describe it?” command. The generated paragraph is divided into two sections: a *description* section and an *evolution* section. The former describes general properties of the graph: what kind of graph it is, the number of variables, what it is about, the title of the axes, etc. Some of this information is directly taken from the graph representation coming from the graphing utility, but some of the information may be inferred as well. *iGraph* has an ontology of concepts about graphs that allows it to disambiguate certain lexical idiosyncrasies, such as writing “yr.” instead of “year” or “01” instead of 2001, etc. The *evolution* section does this.

The *evolution* section of the paragraph presents some interesting characteristics. The first line tells the user what the graph qualitatively looks like: irregular, “looks like an M” or “it’s a bell curve”, etc. The evolution sentences themselves carry out some degree of graph summarization. If there are more than two increases, decreases or plateaus in sequence, then the program takes them to be a *steady* increase, decrease or plateau, making it more readable, with less points, but, as in all summarization techniques, losing some information on the way. In any case, the lost information may be queried at a later stage by, for instance “what is the value of point 4?”.

Several predicates that allow for querying more “visual” characteristics of these graphs have also been implemented. Three of these predicates are of particular interest because of their psychological underpinnings: *moderate*, *slight* and *sharp* are predicates that help summarize the mathematical

slope (the angle) of the increase or decrease of a particular segment. Obviously, the slope level of segments can be conveyed numerically, informing, for instance, that “Grade 1 increases from 2000 to 2001 with a slope of  $x$ ”, where  $x$  is some real number. In these cases, information that is too specific tends to be less informative in discourse than higher-order concepts (such as slight, moderate, etc) at the time of analyzing, describing and reasoning about line graphs. The issue is much like calling several hues of red ‘red’, rather than giving the specific wave length of that particular color.

Currently, all the information available in a given graph is amenable for querying, type of graph, legends, the value of particular points or segments, the position of different legends, the scale, whether the axes have tick marks, etc.

## Conclusions

To conclude, we have presented *iGraph*, a software application that implements a general purpose dialogue architecture developed at the HOTLab. *iGraph* takes an XML representation of a graph and allows a user to interact by it by means of a natural language interface. This kinds of applications are just a small step towards accessible information for all.<sup>1</sup>

## Acknowledgements

This paper was written with the generous support of ORNEC and Cognos Inc. to L. Ferres, Cognos-NSERC Industry Research Chair to G. Lindgaard (RICSA 234088-05). We would like to thank Antoine Chretien, Martin Lachance, Louis Boucher and Jing Liu for valuable support.

## References

- Ault, H. K.; Deloge, J. W.; Lapp, R. W.; Morgan, M. J.; and Barnett, J. R. 2002. Evaluation of long descriptions of statistical graphics for blind and low vision web users. In *ICCHP ’02: Proceedings of the 8th ICCHP*, 517–526. London, UK: Springer-Verlag.
- Ferres, L.; Parush, A.; Li, Z.; Oppacher, Y.; and Lindgaard, G. 2006a. Representing and querying line graphs in natural language: The iGraph system. In *SmartGraphics 2006*, number 4073 in Lecture Notes in Computer Science, 248–253. Berlin Heidelberg: Springer-Verlag.
- Ferres, L.; Parush, A.; Roberts, S.; and Lindgaard, G. 2006b. Helping people with visual impairments gain access to graphical information through natural language: The igraph system. In *Proceedings of the 10th ICCHP*, Lecture Notes in Computer Science. Springer-Verlag.
- Mittal, V. O.; Carenini, G.; Moore, J. D.; and Roth, S. 1998. Describing complex charts in natural language: a caption generation system. *Computational Linguistics* 24(3):431–467.
- Seneff, S.; Hurley, E.; Lau, R.; Pao, C.; Schmid, P.; and Zue, V. 1998. Galaxy-ii: A reference architecture for conversational system development. In *Proc. ICSLP-98*, volume 3, 931–934.

<sup>1</sup>For a more detailed version of this paper see <http://alba.carleton.ca/papers/flairs07x.pdf>