Development of Spoken Language User Interfaces: A Tool Kit Approach

Hassan Alam, Ahmad Fuad Rezaur Rahman, Timotius Tjahjadi, Hua Cheng, Paul Llido, Aman Kumar, Rachmat Hartono, Yulia Tarnikova, and Che Wilcox

Human Computer Interaction Group, BCL Technologies Inc.
990 Linden Drive, Suite #203, Santa Clara, CA 95050,USA
fuad@bcltechnologies.com

Abstract. This paper introduces a toolkit that allows programmers with no linguistic knowledge to rapidly develop a Spoken Language User Interface (SLUI) for various applications. The applications may vary from web-based e-commerce to the control of domestic appliances. Using the SLUI Toolkit, a programmer is able to create a system that incorporates Natural Language Processing (NLP), complex syntactic parsing, and semantic understanding. The system has been tested using ten human evaluators in a specific domain of a web based e-commerce application. The evaluators have overwhelmingly endorsed the ease of use and applicability of the tool kit in rapid development of speech and natural language processing interfaces for this domain.

1 Introduction

Automatic Speech Recognition (ASR) technology is making significant advancements, and voice recognition software is becoming more and more sophisticated. However, because computers are unable to understand the meaning of the words they identify, the practical use of this technology is severely restricted. This paper describes a Spoken Language User Interface (SLUI) Toolkit that allows programmers to rapidly develop spoken language input for computer applications. The Toolkit will allow programmers to generate SLUI front-ends for new and existing applications by a program-through-example method. The programmer will specify a set of sample input sentences for each task, and the SLUI Toolkit will generate a large set of semantically equivalent sentences. The programmer will then select the sentences needed for the task and the SLUI Toolkit will generate code that will take a users spoken request and execute a command on an application. The work reported here is based on a contract awarded by the Advanced Technology Program (ATP) of National Institute of Standards and Technology (NIST) in the USA [1].

T. Caelli et al. (Eds.): SSPR&SPR 2002, LNCS 2396, pp. 339-347, 2002.

2 Overview

As hand-held computers merge with cellular telephones with limited keyboard and pointing capabilities, a user interface that allows spoken language will become the input of choice [2,3]. Current spoken language interface systems mimic menu driven Graphical User Interfaces (GUI). This does not exploit the full power of *naturally spoken* language that allows users to express commands at a higher level of abstraction than is possible with current GUI or command-line interfaces. However, to develop a spoken language interface such as this, the programmer needs to learn a number of different technologies. These include: *Automatic Speech Recognition (ASR),* for transcribing spoken language, *Syntactic Parser***,** for transforming the transcribed text into a Lexical Conceptual Form, *Semantic Analyzer***,** to understand the meaning of the sentence and *Dialog Manager***,** to manage the interaction between the user and the computer. Most computer programmers do not have the knowledge to develop these components. While Commercial Off The Shelf (COTS) ASR systems are available, a programmer needs to understand linguistics and human discourse theory to write an effective SLUI system that incorporates complex syntactic parsing, semantic understanding and dialog. This situation is similar to the pre X-Windows GUI where programmers had to develop custom GUI after learning graphics programming. This clearly hampered adoption of the technology. If anything, linguistic theory is more complex than learning graphical routines. The proposed SLUI Tool Kit will solve this problem.

The underlying technology for Natural Language Processing is being developed for over 30 years. One of the earliest command-and-control like integrated systems developed by Woods [4] used a QLF like formalism. Some of the more recent systems using robust semantic interpretation include the Core Language Engine developed at SRI Cambridge aimed as an interactive advisor system, the Rosetta automatic translation system, the SQUIRREL portable natural language front-end to databases, the Tacitus system developed at SRI International by Jerry Hobbs, et al, the TRAINS system at the University of Rochester for planning a railroad freight system and the Verbmobil system for translation in face-to-face dialogs. At NIST, a spoken language interface to libraries was developed using COTS ASR, and parsers to retrieve from a database.

The computational linguistics field of parsing is mature in terms of available implementations. The main challenges being faced now are robust parsing and increasing coverage of the different parsers. A lot of work is also being done in automatic learning of parser grammar rules based on corpora. Recently shallow parsing methods have received more attention because of their success in text-extraction. But application of shallow-parsing methods to command-and-control-like sentences have not been a focus. There is a body of work on automatically acquiring semantic concepts knowledge through statistical techniques. Knowledge Representation is an old field in Artificial Intelligence. A lot of the current research is focused on building reusable ontology. Also lexical semantics deals with better representations of the lexical knowledge itself.

Most of the current research on generation focuses on how to make efficient, large-scaled feature based generation systems for dialogs. FUF/SURGE and PENMAN/KPML are two of the most widely used systems.

The Agents field has received widespread interest from Computational Linguists in recent years. Natural Language systems are being built as Agent interface technologies. A good overview of the field can be found from the proceedings of the annual conference on autonomous agents. Although agents architecture is receiving attention, most of the current commercially available production systems are implemented as client-server solutions.

A Spoken Language User Interface (SLUI) is the newest method for controlling a computer program. Using simple Natural Language (NL) input, a user can interact with a program through a SLUI. The SLUI Toolkit is a suite of applications and tools that allow programmers to develop SLUI enabled programs that can process NL input. Using the Toolkit, an NL interface can be incorporated into a new or existing program.

3 Methodology

This Section briefly describes the methodology of the development of SLUI.

- Query Inputter. A GUI with relevant functionality is implemented to easily specify test queries, and to map related application parameters. This is based on [5].
- Syntax Recognizer. A lot of the queries can come with specific types of embedded phrases. Examples of these may include embedded dates, emails, documents, currency and many more. Within specific domains, this list can be a lot longer and richer. We harvest these patterns using regular expression (RE) matching and replace them with standard, pre-selected parameters.
- Spell Checker. At this stage, a Commercial-Off-the-Shelf (COTS) spell checker was used [6].
- Sentence Tokenizer. A sentence tokenizer is implemented to tag various parts of speech (POS).
- Parser. We adopted the MiniPar [7] parser at this stage. We also developed a shallow parser to work as a back-up parser. We implemented a framework where these two parsers can be optimally combined to provide maximum robustness.
- Anaphora Resolver. We implement an anaphora resolver at this stage [8].
- Translator. We implemented a translator module to translate the parsed tree to a semantic tree using an internal data structure. This translator transforms the parse tree into a semantic representation that captures its head, its arguments and all its modifiers and complements.
- Frame Generator. We implemented a frame generator to generate a semantic internal representation of the translated tree.
- Frame Handler. We implemented a module to validate the generated frame.

4 Functionality of SLUI Tool Kit

The SLUI Toolkit assists programmers in creating SLUI enabled programs that recognize NL input. Essentially, The SLUI Toolkit creates a channel of communication between the user and the program through a SLUI. The Toolkit handles syntax and semantic processing with minimal direction, and removes the need for an in depth understanding of linguistics and human discourse theory (although basic understanding of English grammar is helpful) [9]. Using the SLUI Toolkit, a programmer is able to create a system that incorporates Natural Language Processing (NLP), complex syntactic parsing, and semantic understanding.

The SLUI Tool Kit works in the following steps:

- Toolkit begins to create a SLUI by using NLP to create semantic representations of sample input sentences provided by the programmer.
- These representations are expanded using synonym sets and other linguistic devices, and stored in a Semantic Frame Table (SFT). The SFT becomes a comprehensive database of all the possible commands a user could request a system to do.
- The Toolkit then creates methods for attaching the SLUI to the program.
- When the SLUI enabled program is released, a user may enter a NL sentence. The sentence is translated into a semantic frame, and the SFT is searched for an equivalent frame. If a match is found, the program executes the action linked to this frame.

Fig. 1. shows a schematic of the SLUI Tool Kit. The SLUI, Program, User and Programmer are the basic components of the system, and each plays a necessary role in the system as a whole. In this system, the user provides SLUI with relevant inputs, and provides responses to dialog questions. The programmer provides SLUI with setup information. SLUI then uses this information and executes code relevant to the desired action.

5 SLUI Toolkit System Process

All of the functions of the SLUI Toolkit can be categorized into 4 processes. Each process consists of several operations that must be completed by the programmer or the system in order to create a SLUI enabled program. The first 3 processes (Input, Create SFT, and Debug SFT) are executed in the Toolkit's main UI. Once these 3 processes have been completed, the programmer must tie the system together with several external functions in order to deploy the SLUI enabled program.

Fig. 2. describes the basic purpose of the system processes.

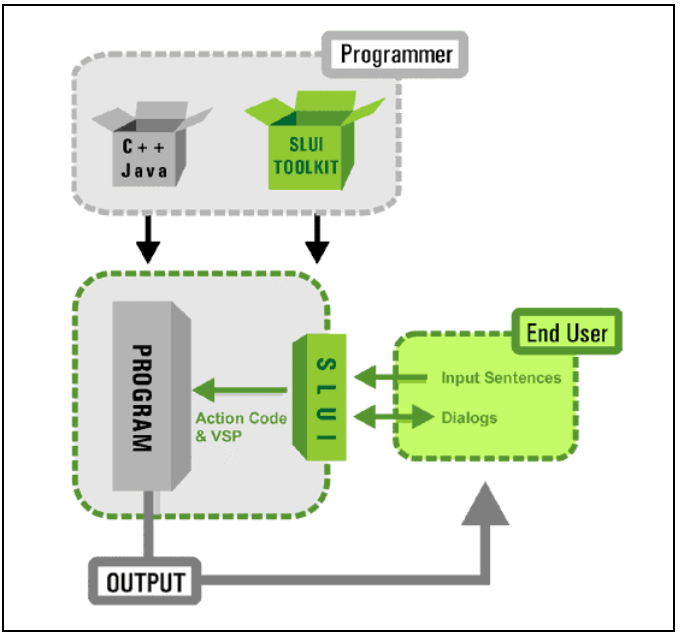

Fig. 1. Component Interaction

5.1 Input Setup Information

The programmer provides the Toolkit with sets of sample input sentences that map to specific actions. The programmer also provides a domain specific lexicon. The Toolkit produces semantic representations of the sample sentences. The SLUI Toolkit needs this information so that it can build a database of all the possibilities the user may input. These are the 3 basic processes within Input Setup Information:

- Provide sample input sentences
  - "When was our last shipment delivered?"
  - "I would like an update on my status."
- Provide lexicon
  - A sample lexicon is provided.
  - The SLUI Toolkit contains tools that can modify this lexicon to fit many different needs.
- Process sample sentences
  - The Toolkit uses the lexicon to extract critical words from the sample input sentences and create semantic representation of them.

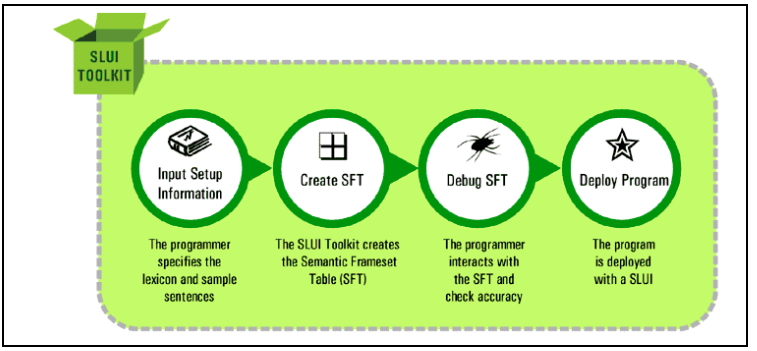

Fig. 2. System Processes

5.2 Create SFT

The SLUI Toolkit uses synonym sets and other linguistic devices to expand the semantic representations of the sample input sentences. These representations are individual frames, and they are stored in a Semantic Frame Table (SFT). The SFT becomes a comprehensive database of all the possible commands, with all possible variations, a user could request a system to do. The programmer then assigns action codes to each frame of the table. These are the 4 basic processes within Create SFT:

- Provide variable sentence parameters (VSP)
  - If the sample input sentence contains a variable, then each variable possibility must be specified:
  - For example: "I would like a <<COLOR>> shirt.", where <<COLOR>> = red, orange, yellow, green, blue, purple etc.
- Expand sample sentences
  - Expand the sentences using synonym sets, Variable Sentence Parameters and the lexicon.
- Assign Action Codes
  - For example, 345: Search for shipment date, 456: Display user status, etc.
- Semantic Frame Table
  - Create the SFT. Fig. 3. SFT structure
  - It shows the structure of a Semantic Frame Table.

Action Code	Sentence Type	Predicate	Argument 1	Argument 2	Argument 3	Head 1	Modifier 1

Fig. 3. SFT structure

5.3 Debug SFT

The programmer may debug the output of the Build Process. It is critical that the sample input sentences have correct semantic representations in the SFT. The programmer does not need a linguistic background to check the values in the SFT. If an SFT element looks as though it would map to an inappropriate User Input sentence, then it should be deleted.

- Analyze SFT
  - The programmer has the opportunity to confirm the results in the Semantic Frame Table.
  - The SLUI Toolkit contains tools that allow the programmer to verify that NL test input will match the proper frames.
- Attach SLUI to Program
  - In order to create a SLUI enabled program, the programmer needs to direct User Input to the SLUI and connect the SLUI to the program.
- COTS ASR Engines
  - Commercial Off The Shelf (COTS) Automatic Speech Recognition (ASR) engines are used to direct User Input to the SLUI.
- Application Agents and Wrappers
  - These utilities assist in binding the SLUI to the C++ code that will be executed in the program.

5.4 Deploy SLUI

When the Toolkit deploys the SLUI enabled program, the user may interact with it. The user enters an NL sentence, and if it has been well anticipated by the programmer, the input sentence maps to the appropriate frame in the SFT. The SLUI sends an action code and the VSP value to the program, and the program executes the appropriate task.

Table 1. Evaluation of SLUI

Subject	Response Scale 1-10
Overall Intuitivity in Using the UI	7.6 / 10
Locating Functionalities	7.5 / 10
Flexibility	6.8 / 10
Modification of Code	7.0 / 10
Integration with an Application	7.6 / 10
VSP Usability	7.5 / 10
Data Organization	8.6 / 10

6 Performance

The system has been tested using ten evaluators. The evaluators have overwhelmingly endorsed the ease of use and applicability of the tool kit in rapid development of

speech and natural language processing interfaces for a web based e-commerce application. Table 1 shows the summary of the evaluation. The basic evaluation was on the usability of the tool kit, how easy it was to integrate with existing applications, how easy it is to write a new interface from scratch and how east it is to modify the interface once it has been designed. The numbers in the second column shows the score in a scale of 1-10, 1 being "poor" and 10 being "excellent".

7 Discussion

The SLUITK is designed to recognize the main concepts in sentences and match User Input sentences with sample sentences provided by the developer. This Natural Language recognition technique works especially well with mono-clausal sentences and common requests. However, currently, the SLUI can only recognize one sense of a word at a time [8]. For example, "Run" can either mean: "Run around the block" or "Run the application." Using a domain specific lexicon will determine which sense of the word is most common. It is critical that the correct sense of a word is chosen to be expanded during the creation of the semantic frame tables. Therefore, we recommend that the programmer verify the results of the Build Process before launching the program.

The biggest advantage of this system is its robustness against variations in the queries made using natural language. A keyword-based approach is easy to develop for a small domain, but as the domain is expanded, the maintenance and updating of such a keyword driven system becomes impossible. SLUI, on the other hand, is very easy to maintain, as the system can be continually updated automatically based on new unseen inputs.

This paper discusses the current status of an ongoing research. We are now working on improving the parsing techniques and automatically grouping similar queries together. Work is also underway to improve automatic rule generation for the parsers and word sense disambiguation for better understanding of the NL inputs.

References

1. Spoken Language user Interface (SLUI) Toolkit. NIST Award #70NANB9H3025.
2. Rahman, A. F. R., Alam, H., Hartono, R. and Ariyoshi, K. Automatic Summarization of Web Content to Smaller Display Devices, 6th Int. Conf. On Document Analysis and Recognition, ICDAR01, USA (2001) pages 1064-1068,
3. Alam, H., Rahman, A. F. R., Lawrence, P., Hartono, R., Ariyoshi, K. Automatic Extraction, Display and Processing of Forms from Web Content to Various Display Devices. U.S. Patent Application pending.
4. Woods, W. Semantics and quantification in natural language question answering. In Advances in Computers. 17(187), (1977).
5. Alam, H. Spoken language generic user interface (SLGUI). Technical Report, AFRL-IF-RS-TR-2000-58, Air Force Research Laboratory, Rome, NY, (2000).

6. Sentry Spelling Checher Engine. Wintertree Software, Nepean, Ontario, Canada K2J 3N4.
7. Scholkopf, B. Dumais, S. T., Osuna, E. and Platt, J. Support Vector Machine. In IEEE Intelligent Systems Magazine, Trends and Controversies, Marti Hearst, ed., 13(4), pages 18-28, (1998).
8. DeKang Lin. University of Manitoba. http://www.cs.ualberta.ca/~lindek/minipar.htm.
9. Mitkov, R. "The latest in anaphora resolution: going robust, knowledge-poor and multilingual". Procesamiento del Lenguaje Natural, No. 23, 1-7, (1998).
10. Burton, A. and Steward, A. P. Effects of linguistic sophistication on the usability of a natural language interface. Interacting with Computers, 5(1), pages 31-59, (1993).