Talkmail

Group ai05-7

Dana Doytch
cl2ddoyt@cling.gu.se

Joakim Evers
evers@etek.chalmers.se

Calle Håkansson
e0calle@etek.chalmers.se

Mattias Sandsäter
mattiass@dtek.chalmers.se

Supervisor: Reiner Hähnle

Artificial Intelligence VT-2005
Chalmers
Gothenburg's University

Abstract

This paper describes an English dialogue system for a mail program, Talkmail. The system has been developed to enable mail users to interact with their mail program by using natural language instead of GUI interaction; even such users who don't necessary have previous experience of natural language processing technology or computers. The aim is to increase the functionality of the program so it could connect to a mail client and allow interaction by speech. In this paper we describe the different parts in a conventional dialogue system and show how we applied these in our system.

Keywords: dialogue, utterance, recognition, natural language processing.

Contents

1. Introduction and purpose

PDAs and other small computers are increasingly difficult to control with input devices such as miniature keyboards or touch screens. People with different disabilities can also take advantage of this system which only uses a speech interface. These devices can easily be replaced by speech controlled devices that allow users to communicate with our programs. For this cause, computers must be able to understand and process natural languages by mapping text to meaning.

Natural language understanding (NLU) has been characterized as a process of hypothesis management in which the linguistic input is sequentially scanned as the system considers alternative interpretations. NLU main concerns include ambiguity, under-specification and ill-formed input.

Our task is to design a reasonable architecture for a specific domain that should work as an additional input device instead of a keyboard, a pen or a mouse. The program should be able to recognize and understand natural languages in form of a text string as its input and should also generate a proper output based on the input interpretation. We choose to implement a dialogue system for a mail domain, assuming that such interface will be essential as many of us daily interact with our mail program.

2. The program design

Our application is thought to enable users to interact with their mail program by natural languages. In order to accomplish this goal we constructed an application that can understand and process natural languages.

The first goal was to clarify how naïve users would interact with a written/spoken dialog system covering the mail domain, which word sequences are common to apply to a desirable command, and what are their possible interpretations. After the first goal was established we could define the vocabulary and construct the suitable grammar which our parser will use to generate the text input from the user.

It was inevitable to set some limitations to the program so we could follow a certain standard, some of which are the following:

- The system always initiates the main dialogue.
- The commands that the system will accept and be able to perform are: receive mail, read mail, reply to a specific mail and compose new mails.
- Every dialogue has a goal which the system will try to achieve.
- One has to finish or abort the current state; it's not allowed to work with two concurrent states.

We defined the different attributes that every command must or may require before it can be completed. For instance the compose command consists of three obligatory attributes that are requested by the system before the command can be executed, namely: sender, subject and message. An optional attribute in this command is attachment.

2.1 Our application

Following our goals and restrictions we designed the following structure for the application as figure 1.

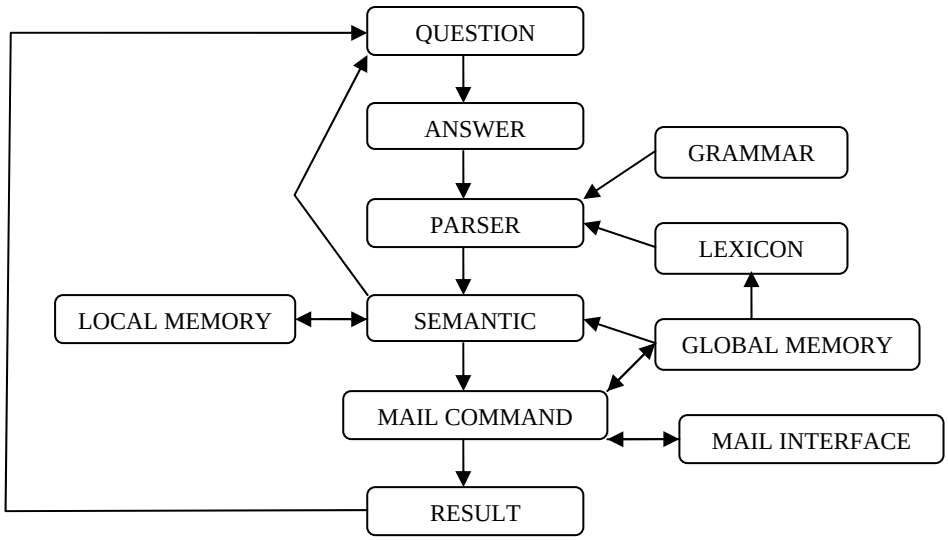

Figure 1: Our application

3. Theory

3.1 Dialogues and dialogue systems

Dialogue is an interactive exchange of meaningful utterances, in this case between a human and a computer using natural language. Dialogue systems make it possible to establish goals and perform computer tasks, such as executing different computer commands like open a certain catalog, search for documents and ask for the time and date.

The basic idea is that every command is followed by a response from the system. The commands typically consist of one major action denoting term plus any number of optional and mandatory flags. The top-level functionality of dialogue systems naturally falls into a sequence of conceptual phases of execution in a fairly standardized way. The following sequence [3.2-3.6] is considered a “standard” sequence of phases when building a domain specific dialogue system.

3.2 Syntactic interpretation

A syntactic interpretation is a linguistic analysis of the user utterance. The syntactic analyzer performs dictionary lookup from the system's lexical source and applies syntactic rules to these word descriptions. The result of this process, which is called syntactic parsing, is declarative descriptions of the syntactic content (subject, object, adverbials, etc.) of each proposed sentence or sentence fragment. This phase typically contains a call to one or more parsers.

3.3 Semantic interpretation

A semantic interpretation is a linguistic analysis of the user utterance. The semantic interpretation produces semantic representations which will be sent to a reasoning module or database which decide the system's reactions. If the input is semantically ambiguous the module will often use statistical knowledge, experience from previous dialogs (case-based learning) in order to choose the most appropriate interpretation. Because it can be appropriate in certain system designs to allow the user to choose among a set of interpretations, the disambiguating effort is optional.

3.4 Pragmatic interpretation

Pragmatic interpretation is refined interpretation of user input based on dialogue context. Typically it is a collection of algorithms using dialogue memory to add/change information in the interpreted structure. When a sentence is parsed and given a semantic interpretation, the relationship between this interpretation and the information previously expressed in the text as well as the interpreter's general knowledge must be established. Establishing this relationship comes under the general heading of pragmatic interpretation. The particular problems that are solved during this step include:

1. Making explicit information that is only implicit in the text. This includes, for example, explicating the relationship underlying a compound nominal, or explicating causal consequences of events or states mentioned explicitly in the text.
2. Resolving anaphoric references and implicit arguments.
3. Viewing the text as an instance of a schema that makes its various parts coherent.

3.5 Task handling and generation

The transformation of an internally represented “system move” to a suitable surface sentence format and/or multimedia-format for the (multimedia) GUI. Typically this phase uses one or more generation filters, e.g. sentence-planning and surface-generation.

3.6 Memorization

The dialogue memory consists of three layers of dialog structure:

1. An intentional structure representing dialogue phases.
2. A thematic structure representing the order in which information has been given.
3. A referential structure keeping track of lexical realizations.

4. The application implementation

The application is implemented in Java 1.5.

Our dialogue system uses a collection of natural language processing modules for English. When the user enters a sentence, it is first processed by a sentence parser, using a combined grammar and a lexicon of English. The result of the parsing is a set of syntactic structures, each associated with a semantic representation of the sentence. The memory module helps to predict (computes) how much of these semantic representations could be incorporated into the current dialogue context. After the semantic module has disambiguated the syntactic structures which are not suitable to use, it will update the dialogue context and decide what to say in response using information from the context and its own private knowledge of facts and goals.

4.1 Dialogues in Talkmail

A message can consist of any possible combination of words and symbols and even non grammatical sentences which will not be accepted by the parser. To solve this problem we decided to incorporate a special sign which marks the message. We choose the hash sign for this and everything between two hash signs is by the parser considered as a single noun.

Dialogue A;1
s - *What do you want to do my lord?*
u - compose a mail
s - *Who do you want to send the mail to?*
u – anna@gu.se
s - *What subject do you want for the mail?*
u - #a meeting on Friday#
s - *What message do you want to send?*
u - #i wanted to remind you about our meeting this Friday#
s - This is the current information:
state = compose
receiver = anna@gu.se
subject = a new meeting on Friday
message = i wanted to remind you about our meeting this Friday
lastquery = execute
s - *Do you want to execute command?*
u - yes
s - *Executing command...*

Dialogue A;2
s - *What do you want to do my lord?*
u - check mail
s - *This is the current information:*
state = receive
lastquery = execute
s - *Do you want to execute command?*
u - yes
Executing command...
You have: 20 new mail(s)
from: anders with subject: Textil och konfektion

…..
from: rune with subject: huller om buller
from: lotta with subject: roliga människor
from: rune with subject: Textil och konfektion

Dialogue A;3
s -*What do you want to do my lord?*
u - reply
s - *What message do you want to send?*
u - #ai project#
s - *This is the current information:*
state = compose
receiver = rune
subject = RE: Textil och konfektion
message = ai roject
lastquery = execute
s - *Do you want to execute command?*
u - yes
s - *Executing command...*
to: rune
from: Reiner
date: Fri Jun 03 18:02:14 CEST 2005
subject: RE: Textil och konfektion
message: ai roject
attachment: null

4.2 Syntactic interpretation

4.2.1 Syntactic Parsing

Parsing can be viewed as a search problem. Two common architectural metaphors are top-down (starting with the root S and growing trees down to the input words) and bottom-up (starting with the words and growing trees up toward the root S). For our purpose we chose to implement the Earley's algorithm, which is a top-down dynamic algorithm. It uses a table of partial parses to efficiently parse ambiguous sentences. The algorithm is considered to be the most efficient possible context-free parsing algorithm, including both the top-down and bottom-up variety.

The items of Earley's algorithm are of the form $[i, A \rightarrow \alpha \bullet \beta, j]$ where α and β are strings in the vocabulary (in our lexicon) and $A \rightarrow \alpha\ \beta$ is a production (a rule) of the grammar. The j index provides the position in the string that recognition has reached, and the dot position marks that point in the partial sentential form. The i index marks the starting position of the partial sentential form, as we localized attention to a single production.

In top-down parsing, we keep a partial sentential form for the material yet to be parsed, using the dot at the beginning of the string of symbols to remind us that these symbols come after the point that we have reached in the recognition process. In bottom-up parsing, we keep a partial sentential form for the material that has already been parsed, placing a dot at the end of the string to remind us that these symbols come before the point that we have reached in the recognition process. In Earley's algorithm we keep both of these partial sentential forms, with the dot marking the point somewhere in the middle where recognition has reached. The dot thus changes from a mnemonic to a necessary role. The algorithm uses a random access

model, include an upper bound on time proportional to n^3, with n standing for the length of string being parsed. It has two major advantages which are:

1. It does not require a special form for the grammar.
2. The algorithm can parse at n^2 given unambiguous input.

4.2.2 Dictionary lookup and grammar rules

Our context-free grammar is simple enough to allow the construction of efficient parsing algorithms which for a given string determine whether it can be generated from the grammar. The grammar consists of a set of rules where every rule contains a left hand-side (a non terminal symbol) and a right hand-side (a sequence of non terminal and terminal symbols). The context-free grammar is important because it is powerful enough to describe the syntax of computer languages. It uses a predefined lexicon which is written for this purpose. In our lexicon we include information about the word class and its interpretation, for instance the verbs: send, compose, write and create have the same interpretation namely, compose.

4.3 Semantic interpretation

Semantic interpretation is the linguistic analysis of meaning. Since a phrase can have several valid meanings, the semantic interpretation is important to distinguish the most probable of them. Our semantic is an operational semantics i.e. a certain action (verb) is defined according to its operational aspects (the attributes required in order for it to operate).
This module uses the dialogue and the global memory to store new information and keep track on the information not yet given. The knowledge stored in the memory is also used to understand the user intention (disambiguate the meaning of an utterance) and to match correctly the input utterance to the different attributes.
The semantic module constantly updates the two memories with new information that is relevant for the current dialogue task.

4.4 Pragmatic interpretation

The phrase coming into the semantic module can have many possible meanings and we want to find the most probable interpretation. In our system the tree structure of the sentence helps us decide on one interpretation which provides most information in the current dialogue, this information is based on the information in the memory.

A state is a mode bound to a specific set of commands, for example "compose mail" and "read mail". Every state has information on what it needs to be complete. When a state is complete it can be executed by a command. The semantic analysis finds out what state it is in by verb keyword search and ask questions to find out the answer to the compulsory information in that specific state.

If the phrase does not contain a verb, it is interpreted as an answer to the last given question. Otherwise, if a verb is found and it is associated with another state, the system asks if the user want to abort the current command. Our semantics does not operate multiple states simultaneous due to the complexity of referenced words. The semantic module always searches for information needed in the current state. The extraction searches for the specific information by analyzing word classes and connecting words that belong to each other. All relevant information to the state is stored in the dialogue memory.

4.5 Command handling and generation

The command module uses a mail interface to connect to the mailbox in which it fetches mails and send mails. Every dialogue has commands bound to it which executes when the

user wants to and the state is complete. The results of the commands are stored in the global memory for further reference.

Due to our mail interface, the application is able to use whatever mail application the user wants by only making an interface to that particular mail application. We have implemented a mail client (Coolmail) which simulates a mail inbox. It continually posts mail to the inbox.

4.6 Memorization

We have two kinds of memories, one dialogue memory and one global memory. The dialogue memory has information about the current state and all information that the state is interested in. The dialogue memory is erased when the user has executed a command. The global memory, on the other hand, is always present and never deleted. It contains references to words like "last mail". We implemented the memory this way so that information will not have to be repeated and anaphoric references could be used during the dialogue. Storing information from previous actions in the dialogue helps to avoid conflicts and decrease ambiguity, so dialogues could interact more naturally.

5. Discussion

With some linguistic background and knowledge about dialogue systems we designed a framework for a dialogue system suitable for a mail interface. Following the standard sequence of phases we constructed a general structure of the system that can be extended and increased. The grammar and lexicon is designed user friendly by means that a certain command is not bound to a specific utterance. A one and same command can be interpreted in different ways by different users. However our natural language interface may not cover the fully desired vocabulary and some users might be irritated by a missing functionality or an unknown word. This problem can be easily overcome by updating the grammar and the lexicon in order to allow various commands and functionalities.

There are some limitations in our semantics; the semantic and pragmatic parts do not resolve anaphoric references. This however does not impose any serious limitations to our program, so we consider it a minor flaw.

The program was developed iteratively; it can easily be increased by adding new functionalities to it.

6. Future work

We hope to increase the program with available commands and connect it to a speech recognizer and speech synthesizer. The goal is to connect this program to a real mail client so it is available for other users and thereby increase its efficiency by manipulating the users perception of the system. Before this can be accomplished, some changes and updates must be considered, some of which are:

1. Increase the program with other commands that exists in mail programs today, such as: add/delete address book, sort messages by name and date.
2. Implement the semantic module so it allows anaphoric references and can deal with bigger amount of commands and requests.

7. References

- Stuart Russell, Peter Norvig; **Second Edition. Artificial Intelligence: A Modern Approach.** Prentice Hall 2003.
- Daniel Jurafsky, James H. Martin: **Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.** Prentice-Hall, 2000. ISBN 0-13-095069-6.

8. Appendix

http://www.etek.chalmers.se/~evers/ai/api/